WELCOM

Thank you very much for choosing our products.
The data package we provide is a zip compression package. Please unzip the data package before learning. Start from this PDF course.

Technical support

If you have any questions about the product, don't worry. You can view the "Common problem.PDF" file or contact us.
Our email is straysnail-wiki@outlook.com
We will reply you within one working day and give you a reasonable solution.

Safety matters

- This product is recommended for children over 6 years old.
- This product needs battery power supply. It is not waterproof. Don't play in water.
- Turn off the power switch when not in use.

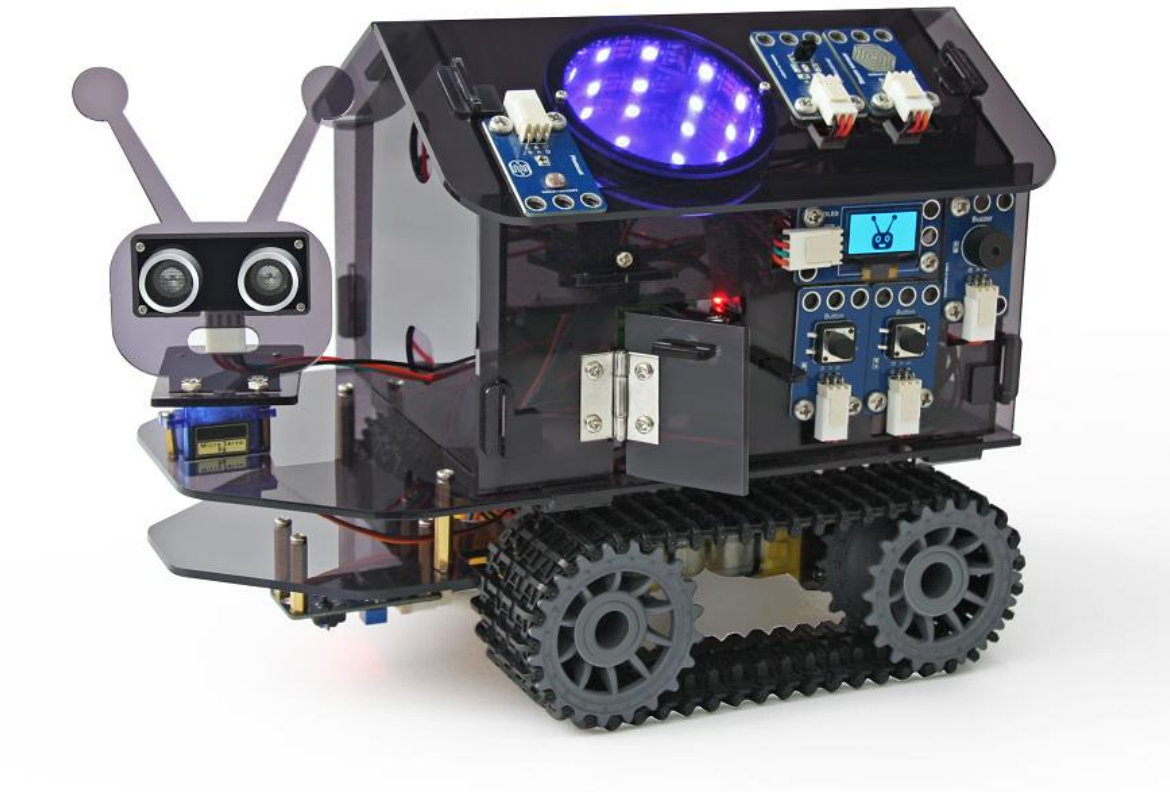
About Stray Snail

- Stray Snail is the brand trademark of Shenzhen Snail Man Intelligent Technology Co., Ltd.
- Mainly for STEAM education products, self developed, including hardware and software design.
- The main products are smart cars, 3D printers and DIY writing plotters, mechanical arms, and some e-learning kits.
- The main control board includes Arduino, raspberry pie, Micro-bit and ESP series.
- Provide customization services: hardware customization, such as the design of sensor modules or the entire product.
- Software customization: Bluetooth and WiFi control APP of Android and iOS, and secondary development customization of scratch3.0 graphical programming.

If you want to know more about Stray Snail, you can visit our official website:

https://www.straysnail.com

Copyright



Ⅰ.The story of Stray Snail

In the quiet and deep forest, a little snail climbs up the huge tree root exposed on the ground and crawls toward the spot of light projected by the sun through the gap between the branches and leaves. This is a snail from the future, wandering around the planet. It has stayed in the forest for several days, and every morning it follows the same route to find sunshine.

It moves slowly, shaking its head from side to side, and equally slowly dodges the obstacles on the road. There is a simple room behind it, where soothing music comes from time to time. On the roof, a lamp ring keeps flashing mysterious colored light, which seems to be transmitting and receiving secrets from the universe. It stops at the light spot and lazily absorbs the heat from the sun.

Gradually, the sky becomes dark, and the water vapor over the forest condenses into rain drops. The open window closes automatically. Until the rain is over, the air is washed clean and clear, the sunlight is transmitted again, and the light column becomes more and more obvious against the background of water vapor.

Suddenly, a light comes on in the room, with the sounds of footsteps. Then, the door is opened 45 degrees outward, and the light in the room comes out from the door. A black shadow also spreads out with the light.

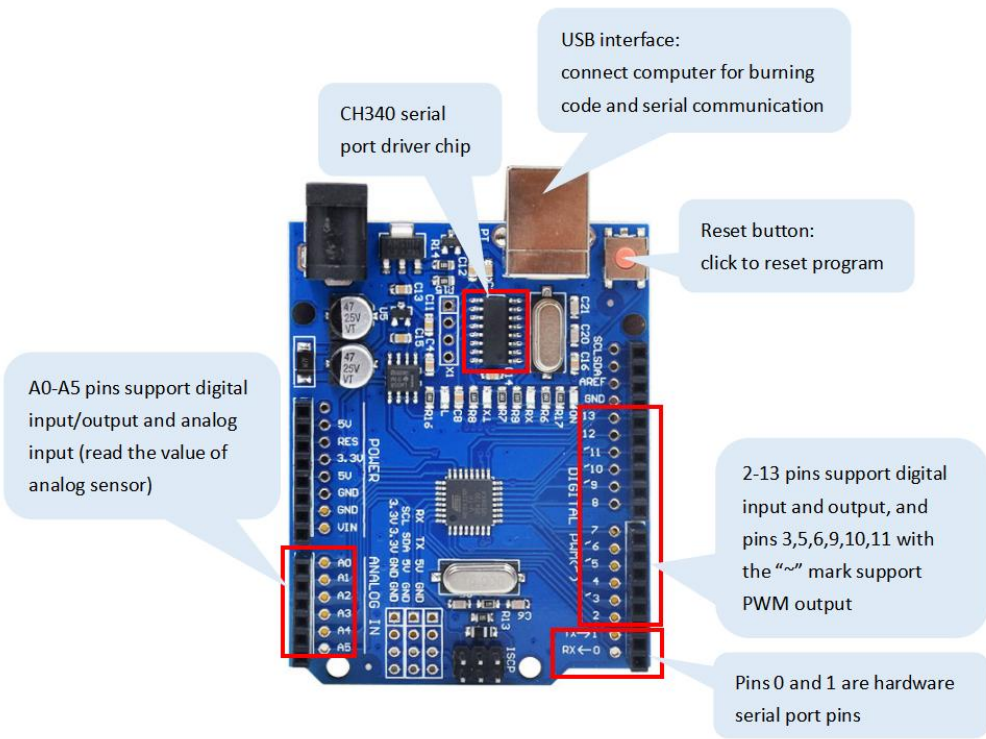
Catalog

Ⅱ. Product features

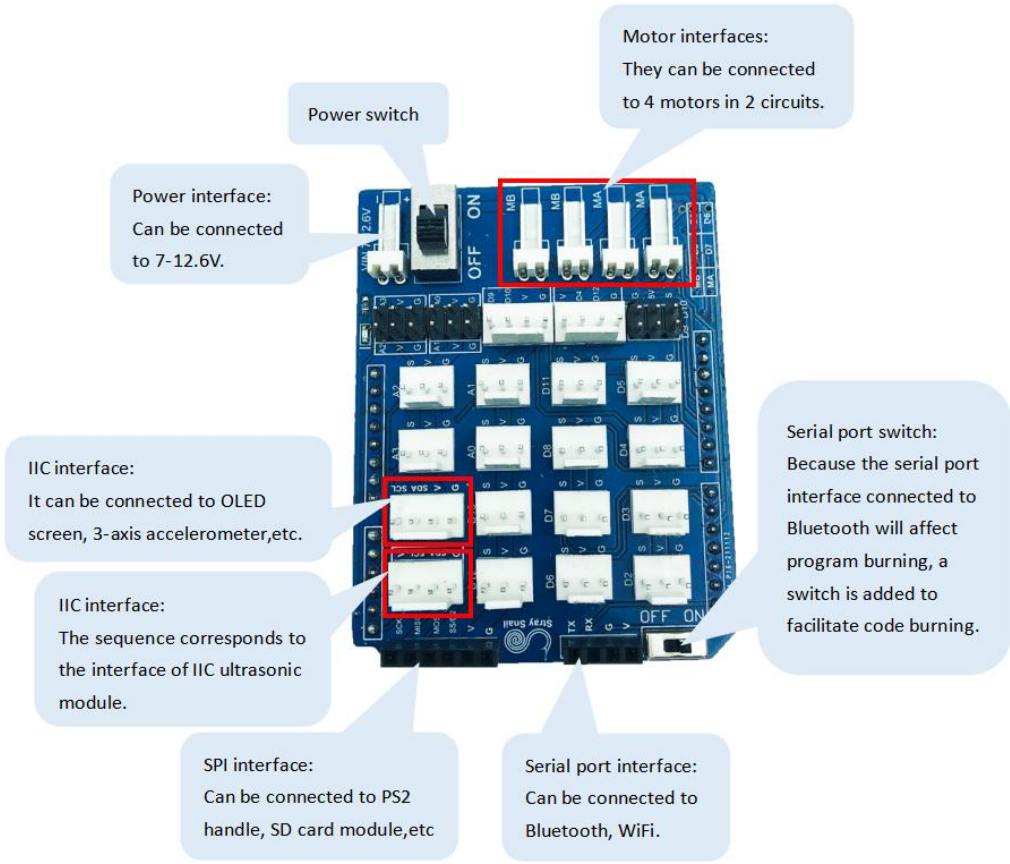
- The structure and appearance are beautiful, and it is also a good artistic decoration on the table.
- There are many kinds of electronic modules, such as digital sensor, analog sensor, driver sensor, etc. Sound, communication and screen, after learning these, you can develop your own products.
- Various and interesting functions, including tracking, following, obstacle avoidance, light tracing, infrared remote control, Bluetooth control, music box, automatic door and window, etc.
- There is a wonderful special effect lamp--the infinite mirror tunnel lamp, which is really cool when it comes on.
- The screen displays multi-function menu, button selection and switching functions, which you can learn to write screen interaction code.

Ⅲ. Products parameters

Input voltage of USB interface of Arduino UNO main board : 5V
Input voltage of external power interface : 7~12.6V
Working voltage of all sensor modules : 5V
The illustration of Arduino UNO main board is as follows :



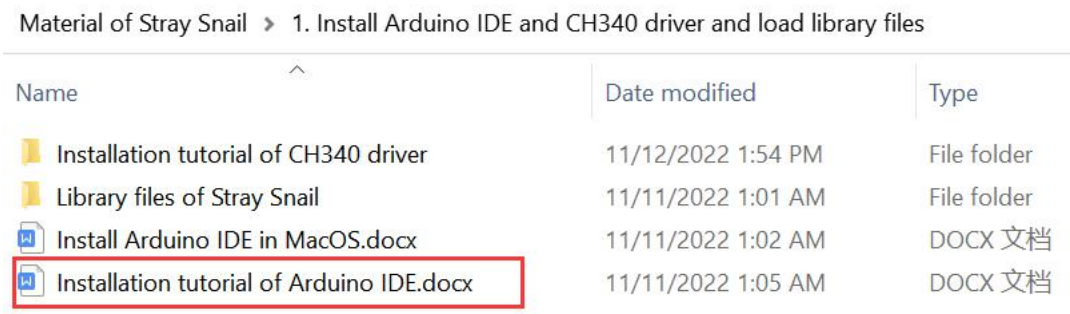
The illustration of expansion board is as follows :



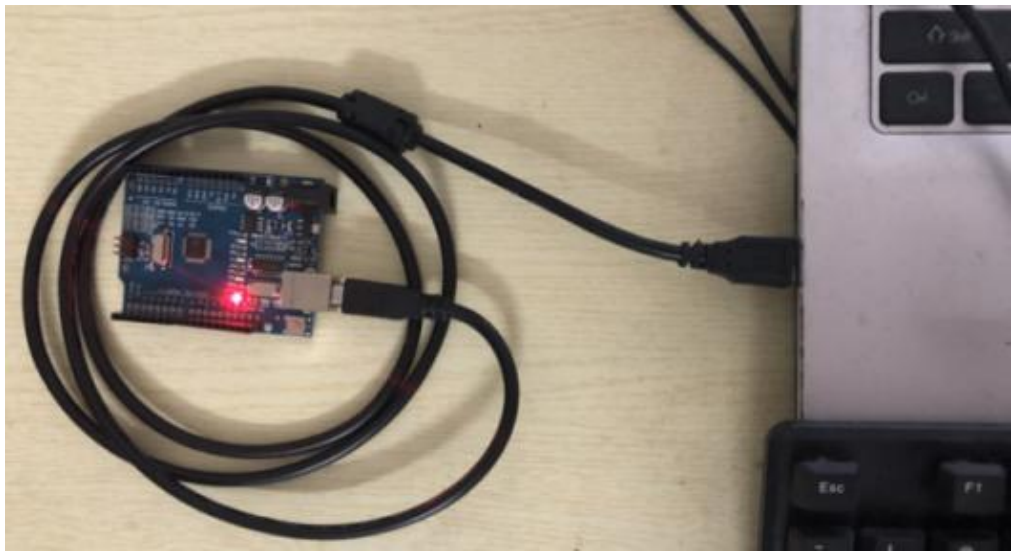
Ⅳ. Install software and its environment configuration

1. Install Arduino IDE and serial port driver

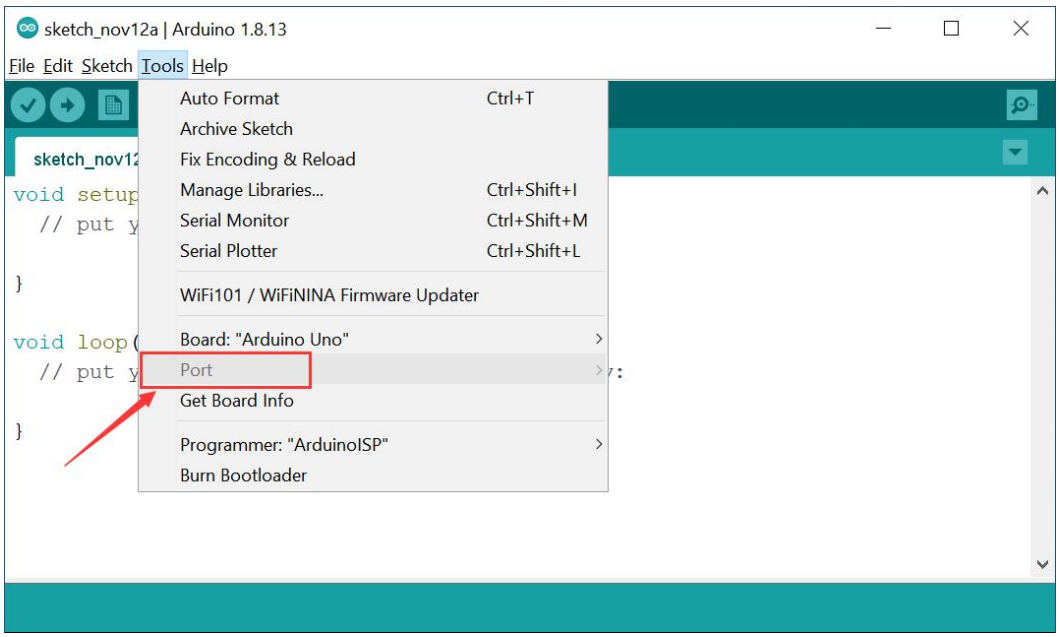
(1) If you are still a novice, please click this link install the Arduino IDE and follow the installation of the Arduino IDE software.



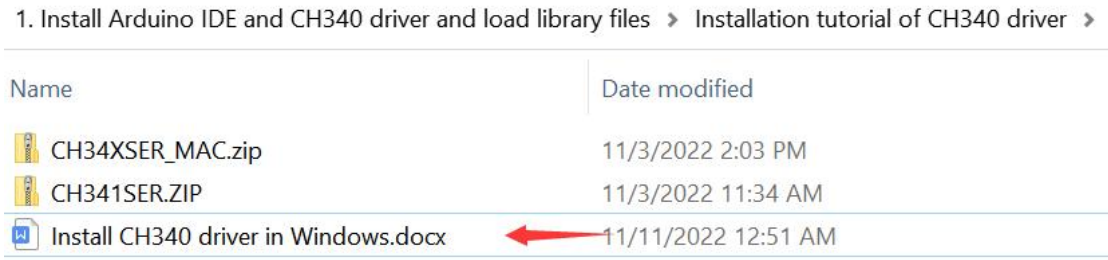
(2) Connect the USB data cable between the Arduino UNO main board and the USB of the computer.



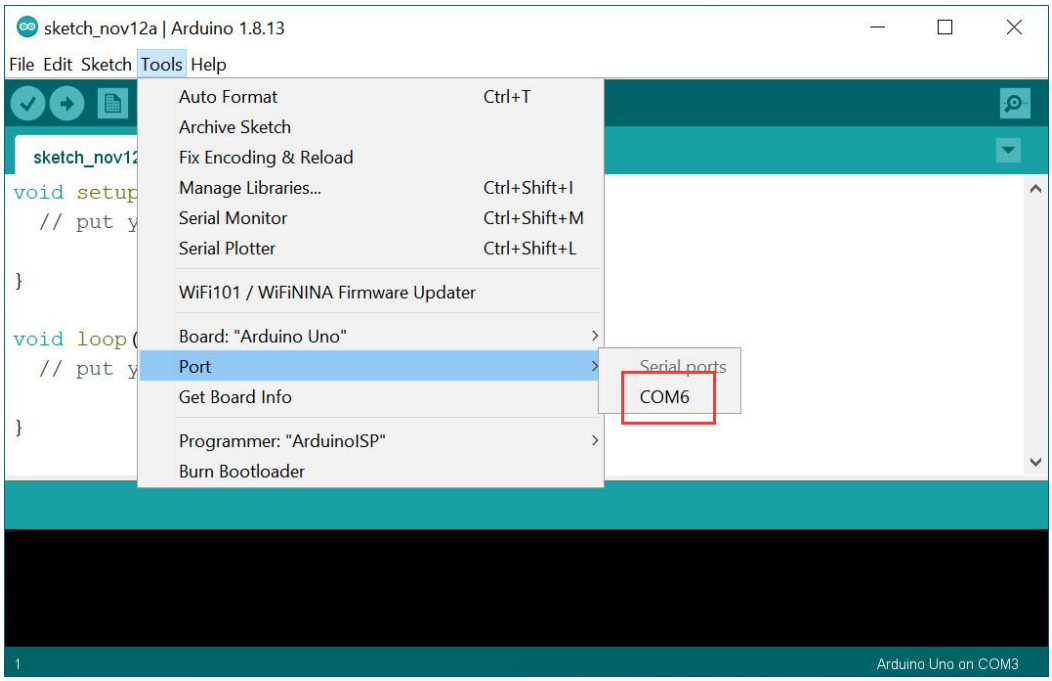
Open the Arduino IDE and click “Tools”. If the CH340 driver cannot be automatically identified, as shown in the following figure, you need to manually install the CH340 driver.



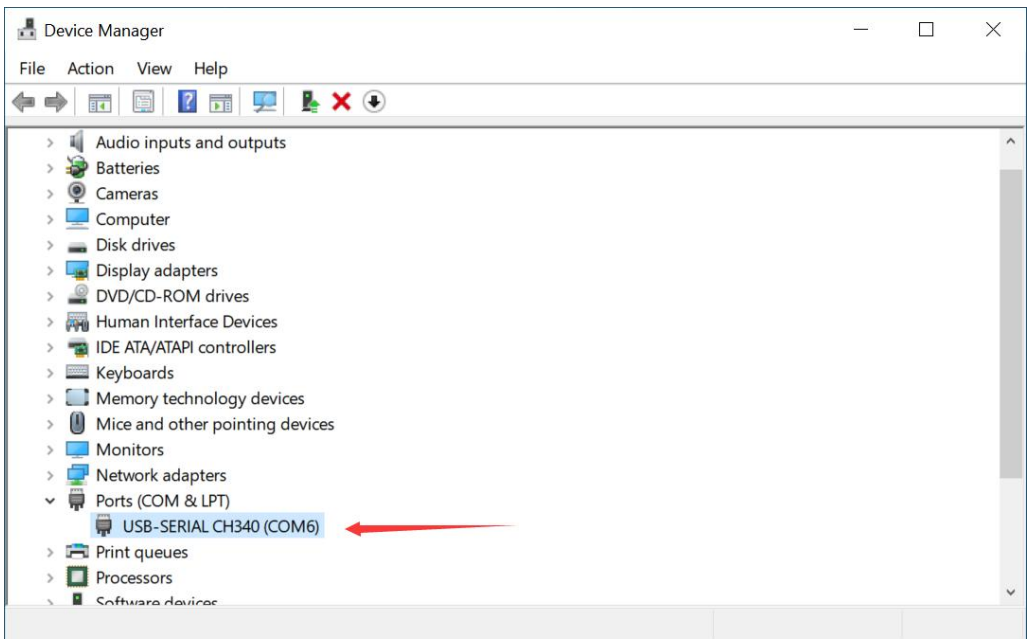
In the material, we provide a CH340 driver installation tutorial. Open the document and follow the steps to install the driver.



If it can be identified, as shown in the figure below.



Open the device manager of the computer to see the CH340 driver, as shown in the following figure.

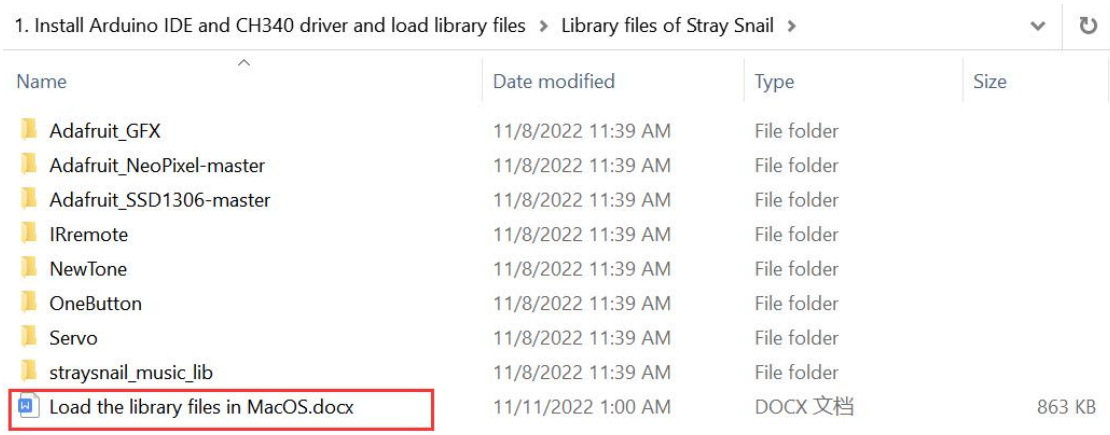


2.Load library file

After installing the software, load the library files provided by us, which will be used in later code learning.

MacOS system loading library：

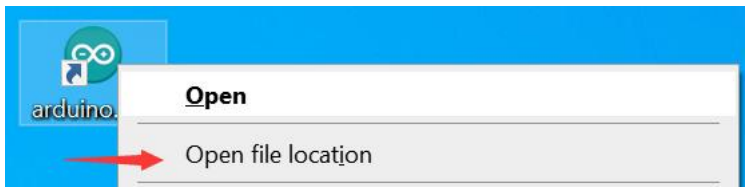
Please open “Load the library files for MacOS”, as shown below.



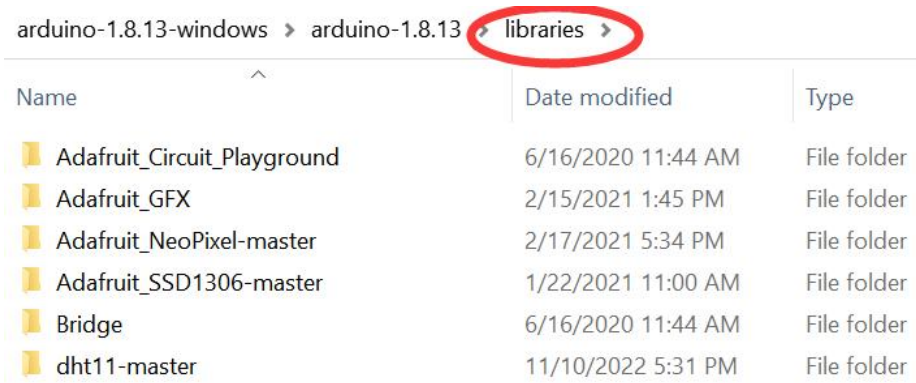
Load the library files for Windows system:

Open the “Stray Snail libraries” folder provided by us, copy all library files, and paste them into “libraries” in the Arduino IDE, as shown in the following figure.

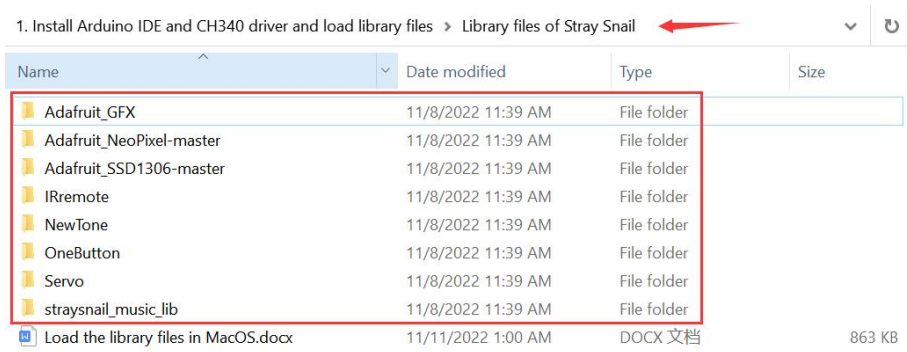
(1) Right click the icon of the Arduino IDE on the desktop and select “open file location”.



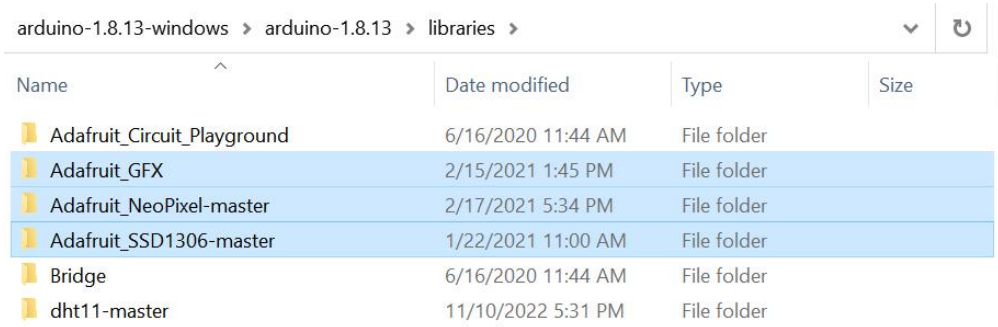
You will enter the installation location of the Arduino IDE and open the “libraries” folder.



(2) Open the library folder provided by us and copy all the library files.



(3) Paste them into the “libraries” folder of the Arduino IDE you just opened. Done.



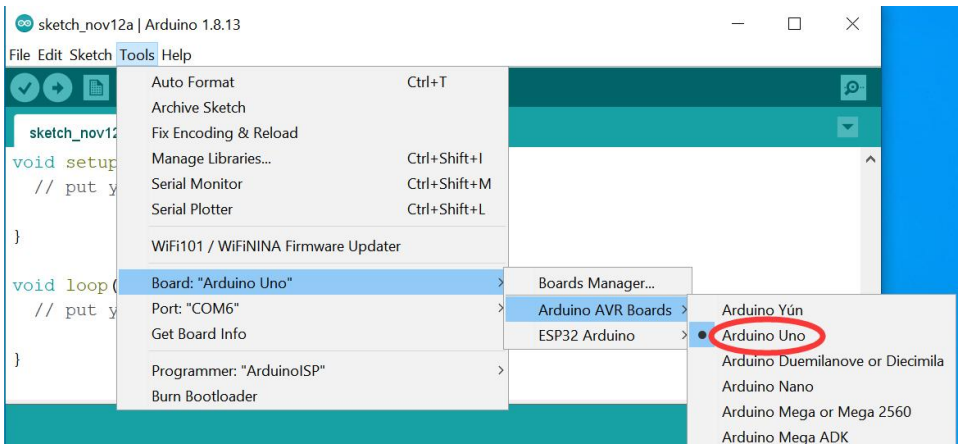
3.Turn on the on-board LED of Arduino IDE

Arduino IDE comes with a lot of example codes. Now we learn to open the code that makes the LED flash in the example and burn it to the Arduino UNO main board.

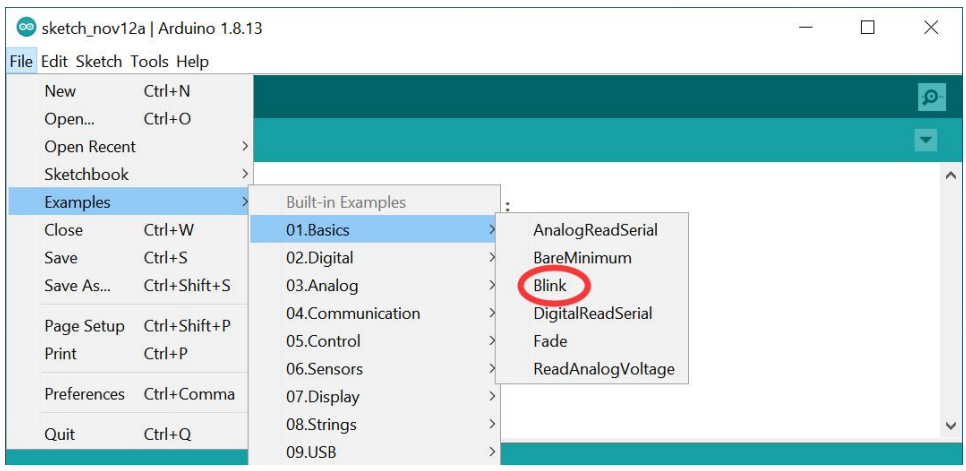
(1) Take out the data cable provided by us, and connect the computer USB and Arduino UNO USB interface.

(2) Open the Arduino IDE and select the main board and COM port in “Tool”

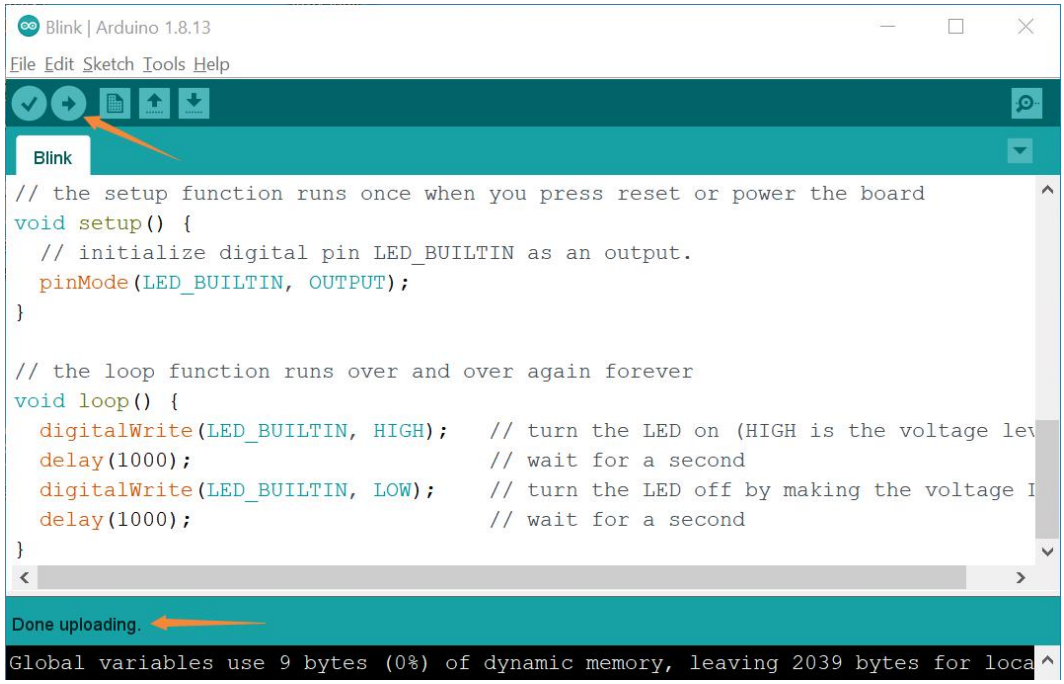
Select “Arduino UNO” as the main board. Pay attention to the COM port. Select the one recognized by your computer, which my computer recognizes is COM4.



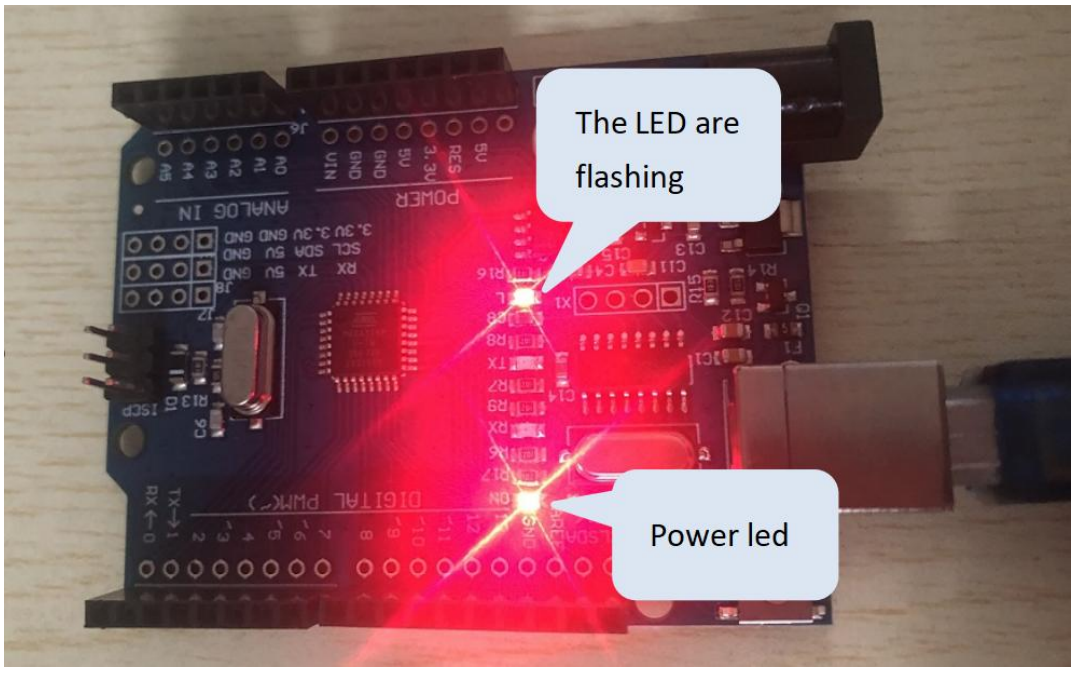
(3) Open the example code coming with the Arduino IDE for turning on the flashing LED.



(4) Click the upload button directly to upload the code to the Arduino UNO main board. If it is successful, you will see that “The upload is successful”.



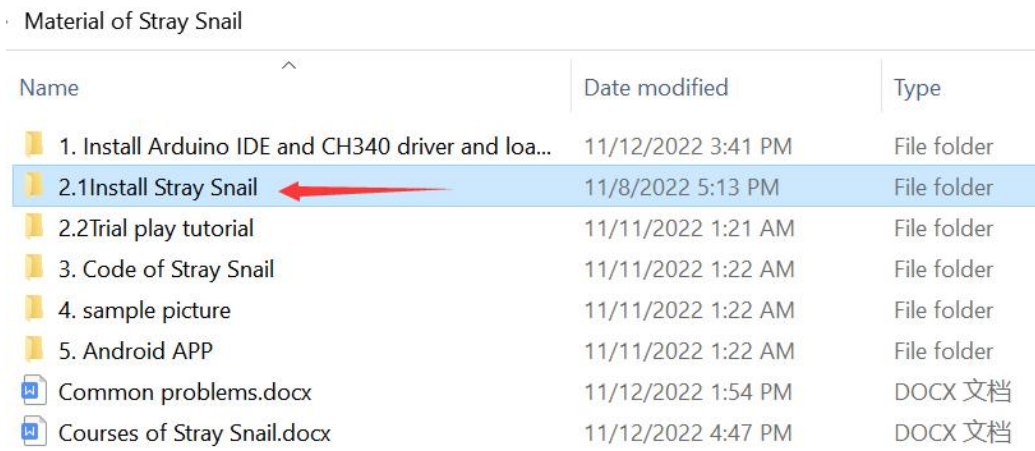
(5) After successful burning, normally you can see the on-board LED of Arduino UNO flashing.



By now, I believe you have a basic understanding of the operation of the Arduino IDE.

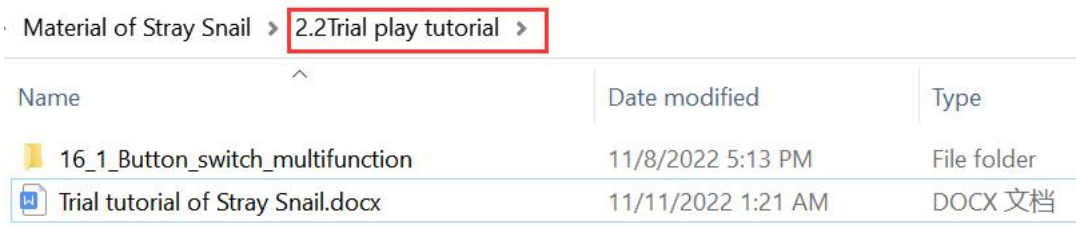
Ⅴ. Assemble the snail

Please open the tutorial provided by us and follow the steps to install the snail.



Power led

After the installation is finished, you may want to play various functions of the snail now. Open the trial tutorial document provided by us.

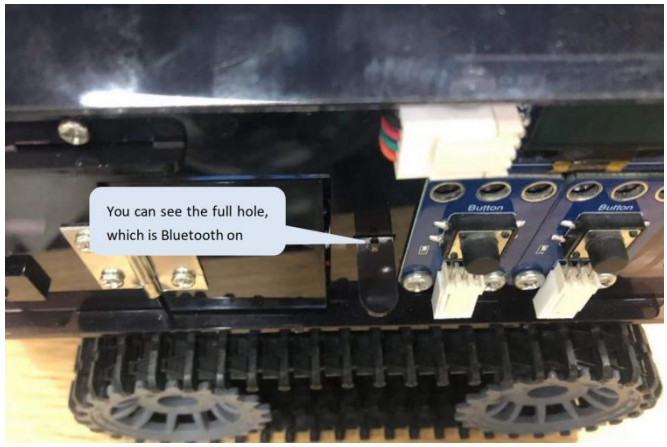


Of course, if you want to learn it step by step and write the code to control the little snail, you can continue to learn code programming.

Ⅵ. Learn to control the Stray Snail

The power switch is turned on by dragging it outward, as shown in the figure below.

Note: The code can be uploaded successfully only when the Bluetooth switch is turned off. The status of Bluetooth off and on is shown in the following figure.



1.Control the LED in the snail house

Story: The night in the forest is very dark, there is no human neon light pollution, and the vast Milky Way in the sky is full of stars. The little snail takes a rest among the huge branches. It turns on the yellow lamp in the house, which flickers from time to time, just like a firefly from afar.

Principle of LED lamp: It is composed of light-emitting diode and protective shell. LED can convert electric energy into light energy.

The IO port of main board or expansion board to connect	A3

(IO is the input and output port.)

1.1 LED flashing

Control the LED to flash like a firefly.

(1) Example code:

```
/*
create by straysnail
2022/2/26
```

```
*/
void setup() {
  pinMode(A3,OUTPUT); //Set A3 pin to output state
}

void loop() {
  digitalWrite(A3,HIGH); //Digital output function, controlling A3 pin output high level
  delay(300); //Delay 300ms
  digitalWrite(A3,LOW);   //Control A3 pin output low level
  delay(300);
}
```

(2) Experimental phenomenon

The LED light in the snail house flashes at an interval of 300ms. The buzzer will sound like a clock when the light is on and off, because the IO port of the buzzer is also connected to A3 (there are too many electronic modules, and the IO ports are not enough).

Tip: It is recommended to modify the value of the delay function, and then observe whether the flashing frequency of the LED light has changed. Remember to reburn the code after modification.

2.Button beside the door

Story: Late at night, the little snail sleeps soundly and has a dream. The picture in the dream is the world it once lived in. When its owner pressed the button on its body, the light immediately came on. It was an order from the master.

Principle of buttons: The buttons installed on the snail are reset buttons. Press down the button cap. After releasing the hand, the button cap will spring up and restore to its original position. It is connected to the circuit as a digital input which can be read in two states. Pressing or not pressing, the corresponding values are 0 and 1.

The button	The IO port of main board or expansion board to connect
Left button 1	D2
Right button 2	D13

2.1 Read the state values of the buttons

Read the state values when two buttons are pressed and released.

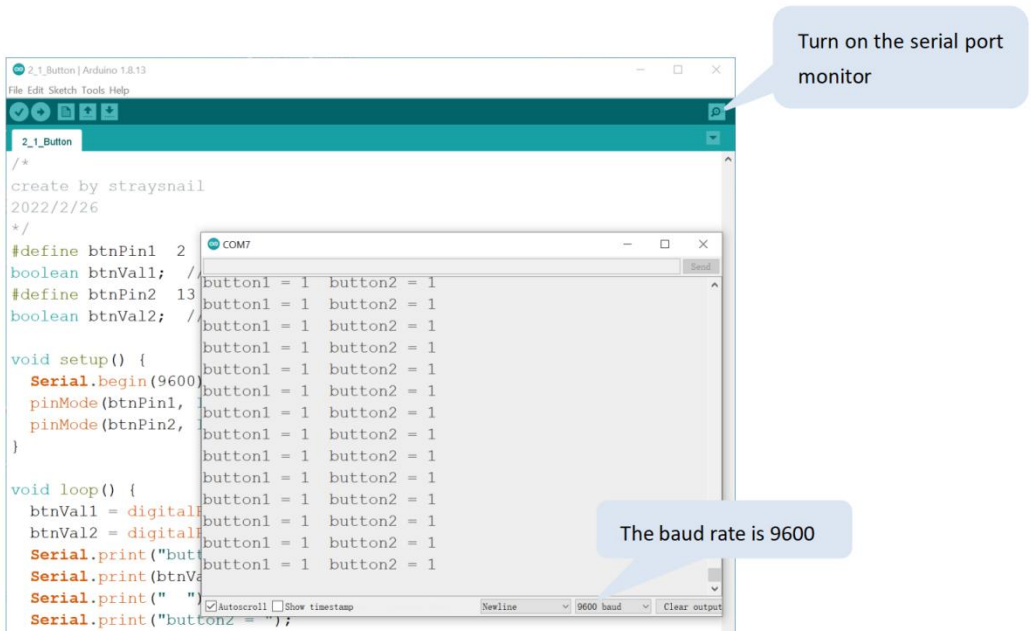
(1) Example code

```
/*
create by straysnail
2022/2/26
*/
#define btnPin1   2     //Macro define the pin name of button 1 as btnPin1, and the corresponding IO port is 2
boolean btnVal1;   //Define a bool type variable to receive the value of button 1
#define btnPin2   13     //13Define the pin of button 2 as 13
boolean btnVal2;   //Define a bool type variable to receive the value of button 2

void setup() {
  Serial.begin(9600);   //9600Set the baud rate of serial communication to 9600
  pinMode(btnPin1, INPUT); //Set button pin as input type
  pinMode(btnPin2, INPUT);
}

void loop() {
  btnVal1 = digitalRead(btnPin1); //Read the value of the button and assign it to btnVal1
  btnVal2 = digitalRead(btnPin2); //Read the value of the button and assign it to btnVal2
  Serial.print("button1 = "); //Serial.print(); function is serial port printing
  Serial.print(btnVal1); //Print the value of button 1
  Serial.print("    ");
  Serial.print("button2 = ");
  Serial.println(btnVal2);   //Serial.println(); function is serial port newline printing
  delay(200);   //The delay is used to adjust the printing speed of the serial port
```

```
}
```

(2) Experimental phenomenon

Open the serial port monitor of the Arduino IDE and **set the baud rate value to 9600** according to the operation shown in the following figure, and then you can see the values printed in the code. Press the button to observe whether the printed value become 0.



Tip: Modify the string printed in “Serial. print()” function and observe whether the string you modified will be printed. The serial port monitor is a very useful tool, which you can directly see the printed values, check where the code stops, etc. You should be good at using the serial port monitor.

2.2 Buttons control the LED lamp

Two buttons, one to control the LED light on, the other to control the LED light off.

(1) Example code

```
/*
  * create by straysnail
  * 2022/2/26
  */
#define ledPin A3    //Define the pin of LED lamp as A3
#define btnPin1 2    //Define the button pin as 2
#define btnPin2 13    //Define the button pin as 13
//The variable is used to receive the value of the button
boolean btnVal1;
boolean    btnVal2;
```

```
void setup() {
  Serial.begin(9600);
  pinMode(btnPin1,INPUT);    //Set the pin to input state
  pinMode(btnPin2,INPUT);    //Set the pin to input state
  pinMode(ledPin, OUTPUT);
}

void loop() {
  btnVal1 = digitalRead(btnPin1);    //Read the value of the button and assign it to btnVal1
  btnVal2 = digitalRead(btnPin2);    //Read the value of the button and assign it to btnVal2
  Serial.println(btnVal2);
  if(btnVal1 == 0)    //If function, and it means that if button 1 is pressed.
  {
    digitalWrite(ledPin, HIGH);    //LED is on
  }
  if(btnVal2 == 0)    //If button 2 is pressed
  {
    digitalWrite(ledPin, LOW);    //LED is off
  }
}
```

(2) Experimental phenomenon

Press the button on the left, the LED lights up, and then press the button on the right, the LED lights out.

Tip: Change “btnVal==0” in the if statement to “btnVal==1”, and then operate and observe.

2.3 Small table lamp

There are many simple table lamps, when press the button, the table lamp will light up, press again, the table lamp will be turned off.

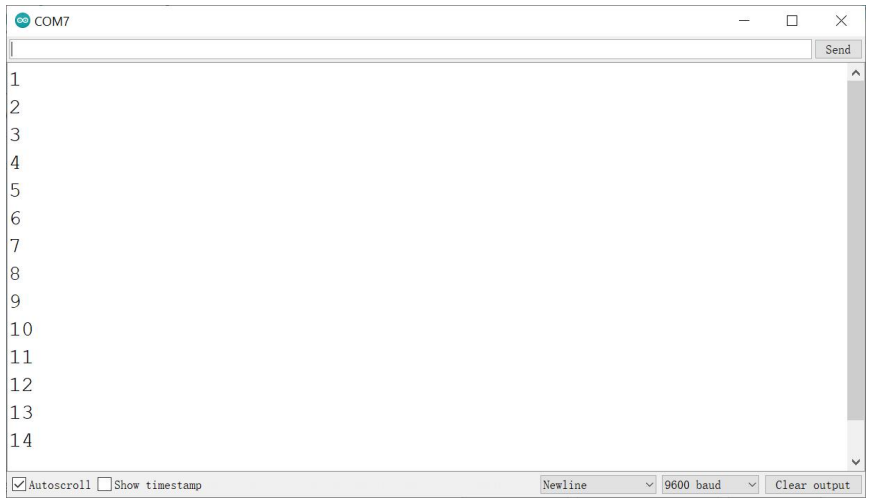
(1) Example code

```
/*
 * create by straysnail
 * 2022/2/26
 */
#define ledPin A3 //Define the pin of LED as A3
#define btnPin 2  //Define the button pin as 2
boolean btnVal;   //Variable, used to receive the value detected by the button
int count = 0; //Variable, used to record the number of times the button is clicked
boolean flag = 0; //Used to switch the state of button press and release
int data;

void setup() {
  Serial.begin(9600);
  pinMode(btnPin,INPUT);  //Set the button pin to input state
}

void loop() {
  btnVal = digitalRead(btnPin);  //Read the value of the button and assign it to btnVal
  if(btnVal == 0)  //If function, and it means that if the button is pressed
  {
    delay(20); //Delay to eliminate the shake of the button
    btnVal = digitalRead(btnPin);
    if(btnVal == 0)
    {
      flag = 1;
      while(flag == 1) //While cyclic function. The cycle will not exit until the button is
released and flag==0
      {
        btnVal = digitalRead(btnPin); //Detect state of the button again
        if(btnVal == 1)  //Judge. If the button is released
        {
          count = count + 1; //Record the number of times the button is clicked
          Serial.println(count);  //Print the number of times the button is clicked
          flag = 0; //Exit the state of button pressing
        }
      }

      data = count % 2;  //Calculate the remainder of the number of clicks on the button to
2. If it is singular, data is equal to 1. If it is even, data is equal to 0
      if(data == 1) //If it is singular
      {
```

```
        digitalWrite(ledPin, HIGH);  //LED is on
      }
      else
      {
        digitalWrite(ledPin, LOW);  //LED is off
      }
    }

  }
}
```

(2) Experimental phenomenon

Press the button on the left once, and the LED lights up. Press the button again, and the LED lights off. Open the serial port monitor of the Arduino IDE, click the button, and you can see that the serial port is printing the number of times the button was clicked.



Tip: In the example code, you can calculate the number of times a button was pressed, and then calculate the remainder of 2 to get 0 or 1. This method is very common. Try writing it yourself several times to get familiar with this usage.

3.Use of buzzer

Story: In the morning, with the fresh air and the surrounding birds' songs and flowers' fragrance, the little snail is in good mood. He hums a song and moves forward slowly.

Principle of passive buzzer: The passive buzzer uses electromagnetic induction to drive the diaphragm to make sound when the electromagnet forms after the voice coil is connected to the alternating current to attract or repel the permanent magnet. It can only make sound when it is connected or disconnected. That is to say, you need to control the frequency of the high and low levels of output to produce sound, and control the frequency to produce different tones.

The IO port of main board or expansion board to connect	A3

3.1 Buzzer sounds

We control the output frequency by controlling the high and low levels and the delay function to make the buzzer sound.

(1) Example code

```
/*
  * create by straysnail
  * 2022/2/26
  */

#define buzPin    A3    //Define the pin of the buzzer as A3

void setup() {
   // put your setup code here, to run once:
   pinMode(buzPin, OUTPUT); //Set to output state
}

void loop() {
   //Set the frequency of the buzzer by delay
   digitalWrite(buzPin, HIGH);
   delayMicroseconds(500);    //Microsecond delay. Within a certain range, adjust the delay, and the buzzer will make different sounds
   digitalWrite(buzPin, LOW);
   delayMicroseconds(500);
}
```

(2) Experimental phenomenon

The buzzer sends out a piercing “Di……”sound.

Tip: Modify the value of the subtle delay function “delayMicroseconds(500)”, and listen to the tones generated by the frequencies of different delay values.

3.2 Use Tone function to control buzzer

It is too troublesome to use high and low level and delay functions to control the tone, so Arduino officially wrote the Tone function using timers, which is convenient for us to write some songs and make music boxes.

(1) Example code

```
#define buzPin A3    //Set the pin of the buzzer as A3
void setup(){
  pinMode(buzPin, OUTPUT); //Set to output state
}

void loop(){
  tone(buzPin,262);    //do
  delay(500);
  tone(buzPin,294);    //re
  delay(500);
  tone(buzPin,330);    //mi
  delay(500);
  tone(buzPin,349);    //fa
  delay(500);
  tone(buzPin,392);    //so
  delay(500);
  tone(buzPin,440);    //la
  delay(500);
  tone(buzPin,494);    //si
  delay(500);
  tone(buzPin,532);    //do
  delay(500);
  noTone(buzPin);        //Stop making sound
  delay(1000);
}
```

(2) Experimental phenomenon

The buzzer emits the sound of “do re mi fa so la si do”.

3.3 Music box

Because the Arduino official Tone function uses timers, while our electronic modules are quite numerous, which will affect the centralized use of multiple functions, we use the “NewTone” library file, which is the same as Tone, but doesn’t need timers. Let’s use the “NewTone” function to write a piece of music.

(1) Example code

```
/*
  * create by straysnail
  * 2022/2/26
  */

#include <NewTone.h> //Import library files for sound frequencies(no timer required)
#define buzzerPin A3      //Set the pin of the buzzer as A3
#define btnPin 2

void setup() {
   Serial.begin(9600);
   pinMode(buzzerPin, OUTPUT);
   pinMode(btnPin, INPUT);
}

void loop() {
   boolean val = digitalRead(btnPin);
   if(val == 0)
   {
      play_music(buzzerPin);
   }
   else
   {
```

```
        NewNoTone(buzzerPin);    //Stop making sound
    }
}

void play_music(int buzzer_pin)
{
    NewTone(buzzer_pin,294,250);      //3 parameters of NewTone: IO port, frequency, duration
    NewTone(buzzer_pin,440,250);
    NewTone(buzzer_pin,392,250);
    NewTone(buzzer_pin,532,250);
    NewTone(buzzer_pin,494,500);
    NewTone(buzzer_pin,392,250);
    NewTone(buzzer_pin,440,250);
    NewTone(buzzer_pin,392,250);
    NewTone(buzzer_pin,587,250);
    NewTone(buzzer_pin,532,500);
    NewTone(buzzer_pin,392,250);
    NewTone(buzzer_pin,784,250);
    NewTone(buzzer_pin,659,250);
    NewTone(buzzer_pin,532,250);
    NewTone(buzzer_pin,494,250);
    NewTone(buzzer_pin,440,250);
    NewTone(buzzer_pin,698,375);
    NewTone(buzzer_pin,659,250);
    NewTone(buzzer_pin,532,250);
    NewTone(buzzer_pin,587,250);
    NewTone(buzzer_pin,532,500);
    NewNoTone(buzzer_pin);
}
```

(2) Experimental phenomenon

Click the button on the left, and the buzzer will play a song of Happy Birthday. When it finishes, you can click to play it again.

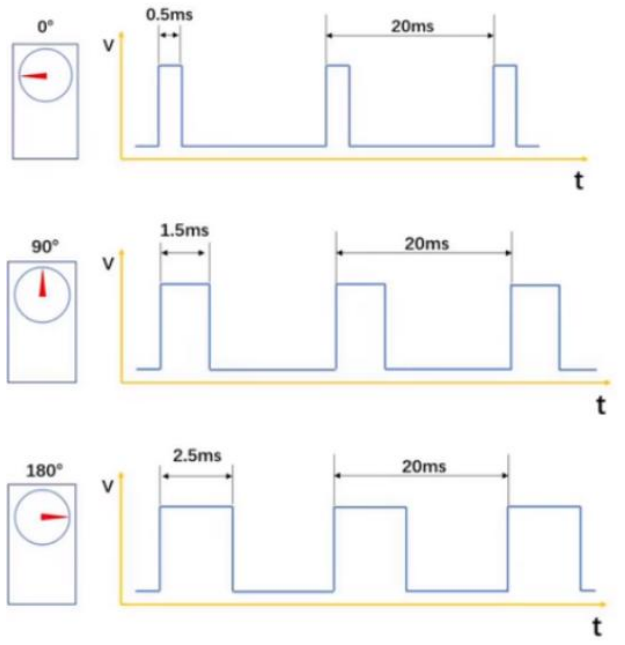
Tip: If you want to write your favorite music, you need to know some theoretical knowledge of music, Understanding the frequency of each tone.

4.Control the steering gears

Story: The little snail climbs up the branch. It feels very comfortable when the wind blows head-on, carrying the fresh fragrance of flowers. It wants to let the wind blow into the

house, so it opens the door and window.

Principle of steering gear control: The angle of the steering gear is determined by the duration of the control signal pulse, which is called pulse code modulation(PCM). The control of the steering gear generally requires a time base pulse of about 20ms. The high level time in the time base pulse is within the range of 0.5ms~2.5ms, and the corresponding angle for controlling the steering gear is 0~180 degrees, as shown in the figure below.



The steering gear is a kind of servo driver, which can control the rotation angle accurately. It is widely and commonly used in the joint movement of robots. The head, door and window of the little snail are controlled by the steering gear, which can rotate 180 degrees.

Steering gear	The IO port of main board or expansion board to connect
Steering gear for the head	D12
Steering gear for the door	A0
Steering gear for the window	D4

4.1 Steering gear rotation

Since the rotation angle of the steering gears of the door and window are limited, we will control the steering gear of the head to rotate to the specified angle firstly.

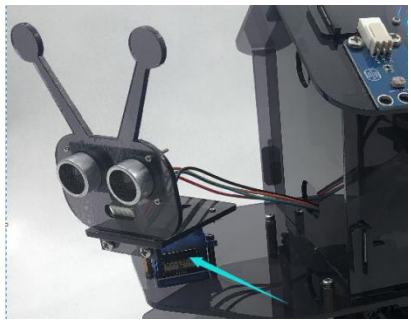
(1) Example code

```
/*
 * create by straysnail
 * 2022/2/26
 */
#define head_ser 12    //Set the pin of the steering gear as 12
```

```
int pulsewidth = 0; //Variable: used to calculate the pulse value of the steering gear

void setup() {
  pinMode(head_ser,OUTPUT); //Set the pin to output state
}

void loop() {
  procedure(head_ser, 0);   //Turn the steering gear to 0 degree
  delay(500);          //Time for the steering gear to turn
  procedure(head_ser, 90);   //90degree
  delay(500);
  procedure(head_ser, 180); //180degree
  delay(500);
  procedure(head_ser, 90);   //90degree
  delay(500);
}

//Function to control the angle of steering gear according to the principle of steering fear
void procedure(int serPin, int myangle)     //There are 2 parameters of the function, serPin is the pin of the steering gear, and myangle is the angle of the steering gear
{
  for(int i=0; i<10; i++)
  {
    //Calculate the pulse value, that is, the high level time. The range of myangle is 0-180, and the corresponding pulse width range is 500-2480
    pulsewidth = myangle * 11 + 500;
    digitalWrite(serPin,HIGH);
    delayMicroseconds(pulsewidth);   //Delay calculated high level time
    digitalWrite(serPin,LOW); //Set to low level
    delay((20 - pulsewidth / 1000));   //Delay the remaining low level time
  }

}
```

(2) Experimental phenomenon

The steering gear rotates from 0°to 90°, then to 180°, and finally to 90°. The cycle is 0-90-180°.



Tip: “void procedure(int serPin, int myangle)” is a sub function in the code. Sub functions are often reused in code. Just write a sub function and call it again, which will make the code look concise. Therefore, be familiar with the use of sub functions.

4.2 Use of Servo steering gear library files

Using servo steering gear library files will make the code more concise and simple, and the internal timer is used to keep the steering gear at the specified angle. First, control the steering gear at the head of the snail to rotate.

(1) Example code

```
#include <Servo.h>    //Import the library files of steering gear
Servo head_servo;    //Create a steering gear instance to control a steering gear

void setup() {
  head_servo.attach(12,500,2500);    //Set the pin of the steering gear as A2, and the pulse range is
500-2500ms
}

void loop() {
  head_servo.write(10);      //The steering gear turns to 10 degree
  delay(500);      //Delay for the steering gear to turn
  head_servo.write(90);    //90 degree
  delay(500);
  head_servo.write(180);
  delay(500);
  head_servo.write(90);
  delay(500);
}
```

(2) Experimental phenomenon

The steering gear rotates from 0°to 90°, then to 180°, and finally to 90°. The cycle is 0-90-180°.

4.3 Control the switch of door and window

The door is controlled by button 1, and the window is controlled by button 2. Attention should be paid to the rotation angle of the steering gears, because the door and window have their own rotation range, and the structure of the door and window may be damaged if the rotation angle is out of the range.

Angle range of the steering gear of the door	90~180°, 90° corresponds to the door closing, and 180° corresponds to the maximum door opening.
Angle range of the steering gear of the window	90~30°, 90° corresponds to the window closing, and 30° corresponds to the window opening.

Note: When writing the code to control the door and window steering gears, try not to exceed the above angle range.

(1) Example code

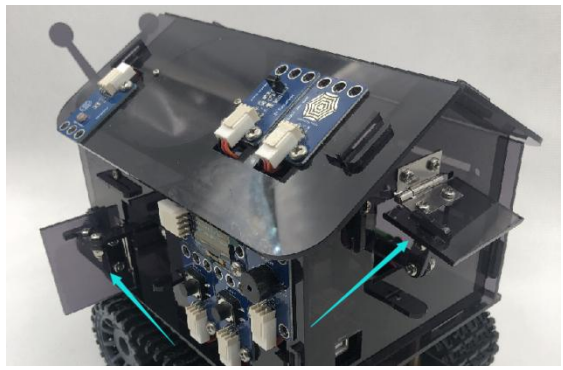
```
/*
 * create by straysnail
 * 2022/2/26
 */
#include <Servo.h>  //Import the library files of steering gear
Servo door_servo;  //Create a steering gear instance to control the door steering gear
Servo window_servo;  //Create a steering gear instance to control the window steering gear
#define btnPin1 2 //IO port of the button
#define btnPin2 13
boolean btnVal1 = 0; //Used to receive the button value
boolean btnVal2 = 0;
int count1 = 0;  //Calculate the click times of the button
int count2 = 0;
boolean door_state = 0; //Control the door state
boolean window_state = 0; //Control the window state

void setup() {
  Serial.begin(9600);
  pinMode(btnPin1, INPUT);
  pinMode(btnPin2, INPUT);
  door_servo.attach(11, 500, 2500);  //Set the pin of the steering gear as 11, and the pulse range is 500-2500ms
  window_servo.attach(4, 500, 2500);  //Set the pin of the steering gear as 4, and the pulse range is 500-2500ms
  door_servo.write(50); //Initial angle is 90 degrees
```

```
  delay(300);
  door_servo.write(140);   //Turn the steering gear to 180 degrees and open the door
  delay(500);   //Delay for the steering gear to turn
  window_servo.write(124);
  delay(300);
  window_servo.write(50);  //Open the window
  delay(500);
  door_servo.write(50); //Close the door
  delay(300);
  window_servo.write(116); //Close the window
  delay(300);
}

void loop() {
  btnVal1 = digitalRead(btnPin1);  //Read the value of the button
  btnVal2 = digitalRead(btnPin2);
  if(btnVal1 == 0)  //If button 1 is pressed
  {
    delay(10); //Delay to eliminate the shake of the button
    btnVal1 = digitalRead(btnPin1);
    if(btnVal1 == 0)
    {
      boolean i1 = 1;
      while(i1 == 1)
      {
        btnVal1 = digitalRead(btnPin1);
        if(btnVal1 == 1)  //Button 1 is released
        {
          count1 = count1 + 1; //Record the click times
          Serial.print("button1 = ");
          Serial.println(count1);
          door_state = count1 % 2; //Calculate the remainder. If it is singular, the value
is equal to 1, and if it is even, the value is equal to 0
          if(door_state == 1)
          {
            door_servo.write(140); //Open the door
            delay(300);
          }
          else
          {
            door_servo.write(50); //Close the door
            delay(300);
```

```
          }
          i1 = 0; //Exit the cycle with i=0
        }
      }
    }
  }

  if(btnVal2 == 0)  //If button 2 is pressed
  {
    delay(10);
    btnVal2 = digitalRead(btnPin2);
    if(btnVal2 == 0)
    {
      boolean i2 = 1;
      while(i2 == 1)
      {
        btnVal2 = digitalRead(btnPin2);
        if(btnVal2 == 1)  //Button 2 is released
        {
          count2 = count2 + 1; //Record the click times
          Serial.print("button2 = ");
          Serial.println(count2);
          window_state = count2 % 2; //Calculate the remainder. If it is singular, the value
is equal to 1, and if it is even, the value is equal to 0

          if(window_state == 1)
          {
            window_servo.write(50); //Open the window
            delay(300);
          }
          else
          {
            window_servo.write(116); //Close the window
            delay(300);
          }
          i2 = 0; //Exit the cycle with i=0
        }
      }
    }
  }
}
```

(2) Experimental phenomenon

If you have initialized the steering gears when installing them, whose phenomenons are to open the door, open the window, close the door, and close the window. Then click the button 1 once to open the door, and click again to close the door. Click the button 2 once to open the window, and click again to close the window.



Tip: The code also uses the method of calculating the number of key clicks and then finding the remainder of 2 to get 0 and 1, which should be consolidated. If the door and window controlled by the above code can not be completely closed, adjust the angle by yourself. The angle should not be too large to avoid damaging the door and window. Remember to turn off the power and unplug the USB cable when problems occur.

5.Rain sensor

Story: The weather suddenly changes, the wind blows the rain clouds over, and the light rain drips on the little snail, so it instinctively closed the door and window. It thinks it is not a thunderstorm, so it takes shelter under the trees.

Principle of rain drop sensor: It is an analog input sensor. The more water drops cover in the detection area, the greater the measured value. Many cars have such sensors to detect rain, and they automatically turn on the wiper to scrape the rain off the windshield.

The IO port of main board or expansion board to connect	A0

5.1 Read the value of the raindrop sensor

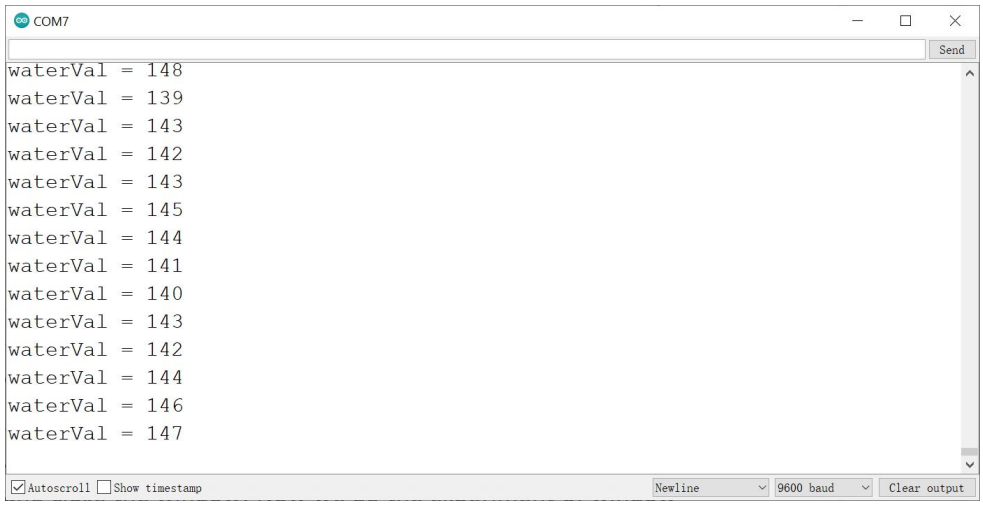
The serial port prints the value detected by the raindrop sensor.

(1) Example code

```
/*
  * create by straysnail
  * 2022/2/26
  */
#define waterPin A0    //Define the IO port of the raindrop sensor as A0
int waterVal = 0;

void setup() {
   Serial.begin(9600);    //Set the baud rate of serial communication to 9600
   pinMode(waterPin,INPUT);    //Set as input
}

void loop() {
   waterVal = analogRead(waterPin);    //Read the analog value detected by the raindrop sensor
   Serial.print("waterVal = ");
   Serial.println(waterVal);    //Print the value detected by the raindrop sensor with new line
   //delay(200); //Delay to adjust the printing speed of serial port
}
```

(2) Experimental phenomenon

Open the serial port monitor of the Arduino IDE, and you can see that the value detected by the raindrop sensor is printed out. Touch the detection area with your hand, and it is found that the value will become larger because there is water on your hand skin. Since it is connected to the same IO port with the LED lamp, you can see that the LED lamp will light up.



5.2 Experiment of window closing when it rains

When I'm outside, I find it is raining. Then I remember that the clothes are hanged outside and the window is not closed. This is a troublesome thing. I think it may be better if I can automatically get the laundry and close the window. Now we do the experiment of window closing automatically when it rains. The principle is very simple. When the rain sensor detects raindrops, the window will be closed automatically.

(1) Example code

```
#include <Servo.h>    //Import the library files of steering gear
Servo window_servo;    //Create a steering gear instance to control the window steering gear
#define waterPin A0    //Define the IO port of the raindrop sensor as A0
int waterVal = 0;

void setup() {
  Serial.begin(9600);    //Set the baud rate of serial communication to 9600
  pinMode(4, OUTPUT);
  pinMode(waterPin,INPUT);    //Set as input
  window_servo.attach(4,500,2500);    //Set the pin of the steering gear as A0, and the pulse range is
500-2500ms
  window_servo.write(116);
  delay(300);
}

void loop() {
  waterVal = analogRead(waterPin);    //Read the analog value detected by the raindrop sensor
  Serial.print("waterVal = ");
  Serial.println(waterVal);    //Print the value detected by the raindrop sensor with new line
  delay(200); //Delay to adjust the printing speed of serial port
  if(waterVal > 100)
  {
    window_servo.write(116); //Close the window
    delay(300);
  }
  else
  {
    window_servo.write(50); //Open the window
    delay(300);
  }
}
```

(2) Experimental phenomenon

At the beginning, the window is open. Then touch the detection area of the rain sensor with your hand, and the window will be automatically closed.

Tip: If you touch the rain sensor detection area and the window is not closed, it may be that your hands are too dry. Try again when touching the water, or debug the value set in the code yourself.

6.Infinite mirror tunnel lamp

Story: After the rain, the little snail suddenly freezes. It seems that it hears something. It quickly climbs up to the height of the tree and opens a light tunnel that seems to be bottomless. Sometimes it changes colors, as if it is receiving electromagnetic signals from the distant universe.

Principle of infinite mirror tunnel lamp: When installing this, you may know something about it. It is very simple. There are two parallel reflectors, and the light from the middle strip light is constantly mirrored between them, which produces an infinite number of luminous lamps. The upper mirror is semi reflective and semi transparent, that is, half of the light can be transmitted and half can be reflected back, so you can see the internal light mirror tunnel.

The IO port of main board or expansion board to connect	A0
LED quantity	11

6.1 Colors of tunnel lamp

There are 256 * 256 * 256=16777216 colors of the WS2812RGB lamp. Let's control the display of several common colors first.

The link of RGB color comparison table: https://tool.oschina.net/commons?type=3

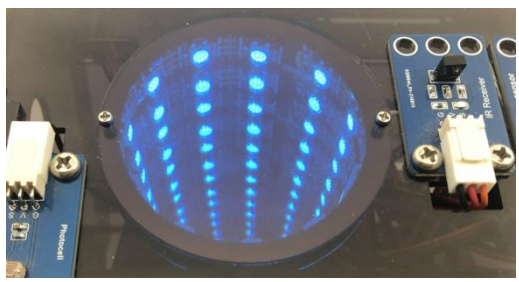
(1) Example code

```
#include <Adafruit_NeoPixel.h>
#define Neo_PIN    A0    //Define the pin as A0
#define NUMPIXELS    11 //Define the LED number as 11
//Declare our NeoPixel strip object
Adafruit_NeoPixel pixels(NUMPIXELS, Neo_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pinMode(A0, OUTPUT);
  pixels.begin(); // Initialize NeoPixel
}

void loop() {
  //Use the for statement to make all LED on
  for(int i=0; i<NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 150, 0));//Set the pin of the LED and RGB color
  }
  pixels.show();//Set the pin of the LED and RGB color

  delay(500);
  pixels.clear(); //Clean up
  for(int i=0; i<NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(150, 0, 0));
  }
  pixels.show();
  delay(500);
  pixels.clear();
  for(int i=0; i<NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 0, 150));
  }
  pixels.show();
  delay(500);
  pixels.clear();
```

```
    for(int i=0; i<NUMPIXELS; i++) {
      pixels.setPixelColor(i, pixels.Color(150, 150, 0));
    }
    pixels.show();
    delay(500);
    pixels.clear();
    for(int i=0; i<NUMPIXELS; i++) {
      pixels.setPixelColor(i, pixels.Color(0, 150, 150));
    }
    pixels.show();
    delay(500);
    pixels.clear();
    for(int i=0; i<NUMPIXELS; i++) {
      pixels.setPixelColor(i, pixels.Color(150, 150, 150));
    }
    pixels.show();
    delay(500);
    pixels.clear();
}
```

(2) Experimental phenomenon

The different colors of the infinite mirror tunnel lamp are switched in cycles.



Tip: The parameter “i” in the code “Pixels.setPixelColor(i, pixels.Color(0, 150, 0));” specifies the lamp number, and the last three parameters are the RGB values of the lamp, whose range is 0~255. Modify these parameters and then observe the lamp. Remember to add the display statement “pixels.show();” To display. You can choose the RGB values according to the RGB color comparison table: https://tool.oschina.net/commons?type=3

6.2 Mirror tunnel breathing lamp

The light is getting brighter and darker. It seems to be breathing.

(1) Example code

```
#include <Adafruit_NeoPixel.h>
```

```
#define Neo_PIN    A0    //Set the pin as A0
#define NUMPIXELS    11 //Define the LED number as 11
//Declare our NeoPixel strip object
Adafruit_NeoPixel pixels(NUMPIXELS, Neo_PIN, NEO_GRB + NEO_KHZ800);
int brightness = 0; //The variable is used to set the the lamp brightness, and the maximum brightness value is 255

void setup() {
  pixels.begin(); //Initialize NeoPixel
}

void loop() {
  //Realize the purple breathing lamp
  for(int bright_val=0;bright_val<256;bright_val++)
  {
    displayPiexls(NUMPIXELS, bright_val, 0, bright_val);
    pixels.show();//Display
    delay(10);
  }
  for(int bright_val=255;bright_val>0;bright_val--)
  {
    displayPiexls(NUMPIXELS, bright_val, 0, bright_val);
    pixels.show();//Display
    delay(10);
  }
}

//Set a function to control all the LED and their brightness
int displayPiexls(int num, int redVal, int greenVal, int blueVal)
{
  //Use for loop statement to make all LED on
  for(int i=0; i<num; i++) {
    pixels.setPixelColor(i, pixels.Color(redVal, greenVal, blueVal));//Set the pin of the LED and RGB color

  }
}
```

(2) Experimental phenomenon

The purple tunnel light, gradually light and gradually dark.

Tip: Modify the code to make the breath lamp emit the color you like.

6.3 Switch lamp colors with a button

(1) Example code

```
/*
 * create by straysnail
 * 2022/2/26
 */
#include <Adafruit_NeoPixel.h>
#define Neo_PIN  A0  //Set the pin as A0
#define NUMPIXELS  11 //Define the LED number as 11
//Declare our NeoPixel strip object
Adafruit_NeoPixel pixels(NUMPIXELS, Neo_PIN, NEO_GRB + NEO_KHZ800);

#define btnPin 2  //Define the button pin as 2
#define btnPin2 13
boolean btnVal1;   //The variable is used to receive the value detected by the button
boolean btnVal2;
int count = 0; //The variable is used to record the click times of the button
boolean flag1 = 0; //Used to switch the state of button press and release
boolean flag2 = 0;

void setup() {
  Serial.begin(9600);
  pinMode(btnPin,INPUT);  //Set pin 11 to input
  pixels.begin(); // Initialize NeoPixel
  pixels.clear(); //Clean up
  pixels.show();//Display
}

void loop() {
  btnVal1 = digitalRead(btnPin);  //Read the button value and assign it to btnVal1
  if(btnVal1 == 0)  //Judge if the button is pressed
  {
    delay(10);
    btnVal1 = digitalRead(btnPin);
    if(btnVal1 == 0)
    {
      flag1 = 1;
      while(flag1 == 1) //When the button is pressed
      {
        btnVal1 = digitalRead(btnPin); //Detect the button state again
```

```
        if(btnVal1 == 1)  //Judge if the button is released
        {
          count = count + 1; //Record the click times of the button
          Serial.println(count);  //Print the click times of the button
          flag1 = 0; //Exit the pressed state
          if(count >= 6)
          {
            count = 6;
          }
          switch(count)
          {
            case 0: displayPiexls(NUMPIXELS, 200, 0, 0); pixels.show(); break;
            case 1: displayPiexls(NUMPIXELS, 0, 200, 0); pixels.show(); break;
            case 2: displayPiexls(NUMPIXELS, 0, 0, 200); pixels.show(); break;
            case 3: displayPiexls(NUMPIXELS, 200, 200, 0); pixels.show(); break;
            case 4: displayPiexls(NUMPIXELS, 0, 200, 200); pixels.show(); break;
            case 5: displayPiexls(NUMPIXELS, 200, 200, 200); pixels.show(); break;
            case 6: displayPiexls(NUMPIXELS, 0, 0, 0); pixels.show(); break;
          }
        }
      }
    }

}
btnVal2 = digitalRead(btnPin2);  //Read the button value and assign it to btnVal1
while(btnVal2 == 0)  //Judge if the button is pressed
{
  delay(10);
  flag2 = 1;
  while(flag2 == 1) //When the button is pressed
  {
    btnVal2 = digitalRead(btnPin2); //Detect the button state again
    if(btnVal2 == 1)  //Judge if the button is released
    {
      count = count - 1; //Record the click times of the button
      Serial.println(count);  //Print the click times of the button
      flag2 = 0; //Exit the pressed state
      if(count <= 0)
      {
        count = 0;
      }
      switch(count)
```

```
        {
          case 0: displayPiexls(NUMPIXELS, 200, 0, 0); pixels.show(); break;
          case 1: displayPiexls(NUMPIXELS, 0, 200, 0); pixels.show(); break;
          case 2: displayPiexls(NUMPIXELS, 0, 0, 200); pixels.show(); break;
          case 3: displayPiexls(NUMPIXELS, 200, 200, 0); pixels.show(); break;
          case 4: displayPiexls(NUMPIXELS, 0, 200, 200); pixels.show(); break;
          case 5: displayPiexls(NUMPIXELS, 200, 200, 200); pixels.show(); break;
          case 6: displayPiexls(NUMPIXELS, 0, 0, 0); pixels.show(); break;
        }
      }
    }
  }
}

//Define a function to control all the LED and their brightness
int displayPiexls(int num, int redVal, int greenVal, int blueVal)
{
  //Use for loop statement to make all LED on
  for(int i=0; i<num; i++) {
    pixels.setPixelColor(i, pixels.Color(redVal, greenVal, blueVal));//Set the pin of the LED
and RGB color
  }
}
```

(2) Experimental phenomenon

Click two buttons to switch the color of tunnel light back and forth.

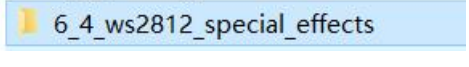
Tip: The code uses the key counting function, as well as “switch(count){case 1: break;}” conditional statements. Compared to “if”, when there are many conditions to judge, use the switch statement to make the code look more concise.

6.4 Special lighting effect

Control the tunnel light to display various special light effects.

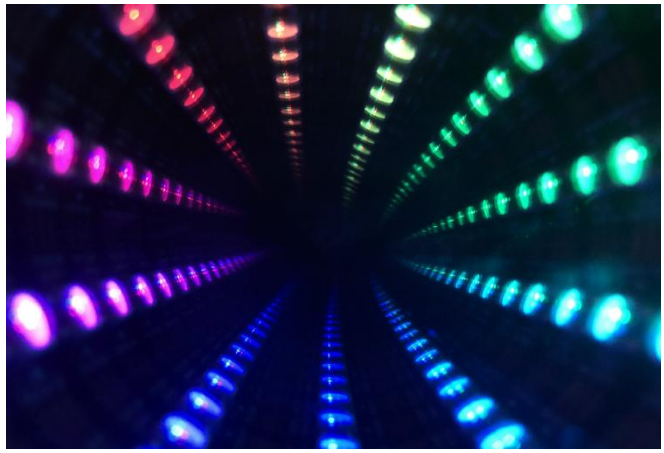
(1) Example code

Because the code is so long, please open the example code to view.



(2) Experimental phenomenon

The tunnel light shows special light effects.



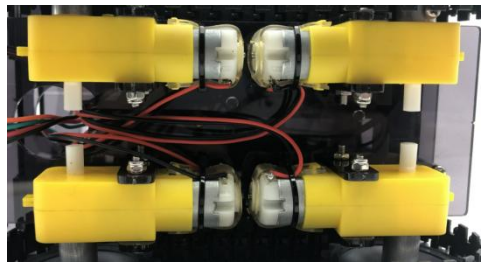
Tip: The code of special light effects is difficult to write. If you are interested in making some atmosphere lights, you should spend some time to understand the code of special light effects.

7.Driving motor

Story: The little snail speeds up and runs forward. It wants to get to the place mentioned in the signal.

Wheels: The wheels of the small snail are composed of 4 motors, 4 driving wheels and tracks. It has sufficient power, strong traction of the tracks, and more importantly, it looks cool!

Drive motor: The driving chip of the motors is TB6612, which is integrated on the expansion board. We have made some adjustments to its peripheral circuit, so that it can control two circuits of motors well with 4 IO ports instead of the original 7, reducing the occupation of IO. You should know that when the product is large, the IO ports of Arduino UNO are very valuable.



Control pin of left motor	Control pin of right motor

The IO port to control direction: D7	The IO port to control direction: D8
The IO port to control speed: D6	The IO port to control speed: D5

7.1 Control movements of the little snail

It is very simple to control the movement of the little snail. The IO port of controlling direction outputs high level or low level, the IO port of controlling speed outputs the analog value 0~255, which can control the forward or backward speed of the motor.

Left motor		Right motor		Direction
Directional IO port: **HIGH**	Speed IO port: 30~255	Directional IO port: **HIGH**	Speed IO port: 30~255	**Forward**
Directional IO port: **LOW**	Speed IO port: 30~255	Directional IO port: **LOW**	Speed IO port: 30~255	**Back**
Directional IO port: **LOW**	Speed IO port: 30~255	Directional IO port: **HIGH**	Speed IO port: 30~255	**Turn left**
Directional IO port: **HIGH**	Speed IO port: 30~255	Directional IO port: **LOW**	Speed IO port: 30~255	**Turn right**
Directional IO port: **LOW**	Speed IO port: **0**	Directional IO port: **LOW**	Speed IO port: **0**	**Stop**

(1) Example code

```
/*
  * create by straysnail
  * 2022/2/26
  */
#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to Control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to Control the Define pin 6 to Control the speed of left motorspeed of right motor

void setup() {
   pinMode(INA, OUTPUT);    //Set the pins controlling the motor to output
   pinMode(ENA, OUTPUT);
   pinMode(INB, OUTPUT);
```

```
    pinMode(ENB, OUTPUT);
}

void loop() {
    car_forward();    //Forward
    delay(2000);      //2s
    car_back();       //Backward
    delay(2000);      //2s
    car_left();       //Turn left
    delay(2000);      //2s
    car_right();      //Turn right
    delay(2000);      //2s
    car_stop();       //Stop
    delay(1000);      //1s
}

//The little snail goes forward
void car_forward()
{
    digitalWrite(INA, HIGH);    //Output high level and control the left motor forward
    analogWrite(ENA, 100);      //PWM output to control the speed of left motor
    digitalWrite(INB, HIGH);    //Output high level and control the right motor forward
    analogWrite(ENB, 100);      //PWM output to control the speed of right motor
}
//The little snail moves backward
void car_back()
{
    digitalWrite(INA, LOW);    //Output low level and control the left motor backward
    analogWrite(ENA, 100);     //PWM output to control the speed of left motor
    digitalWrite(INB, LOW);    //Output low level and control the right motor backward
    analogWrite(ENB, 100);     //PWM output to control the speed of right motor
}
//The little snail turns left
void car_left()
{
    digitalWrite(INA, LOW);
    analogWrite(ENA, 100);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 100);
}
//The little snail turns right
void car_right()
```

```
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 100);
  digitalWrite(INB, LOW);
  analogWrite(ENB, 100);
}
//Stop
void car_stop()
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 0);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

The little snail moves forward, backward, turns left, turns right, and stops.

Tip: Modify the analog output value "analogWrite (ENA, 100);", which range is 0~255, changing the speed of the little snail. If the PWM value is too small, the power will be insufficient. It is recommended to write more than 30.

8.Tracking of the little snail

Story: There is a path on the grass, on which the black soil hasn't been turned up for a long time. The little snail thinks that the objects passing by are not far away, and it should be able to follow along the black soil path.

Principle of the Two tracking sensor circuits: The probe of the tracking sensor is a pair of infrared diode, one emitting infrared and the other receiving infrared. Within a certain distance, when the tracking sensor is aimed at a white object (bright object), the infrared ray emitted will be transmitted back and received by the receiving diode. The black object will absorb light, so when the emitting diode is aimed at the black object, the infrared ray will be absorbed, and the receiving diode will not receive the infrared ray.

The IO port of left tracking	D9
The IO port of right tracking	D10

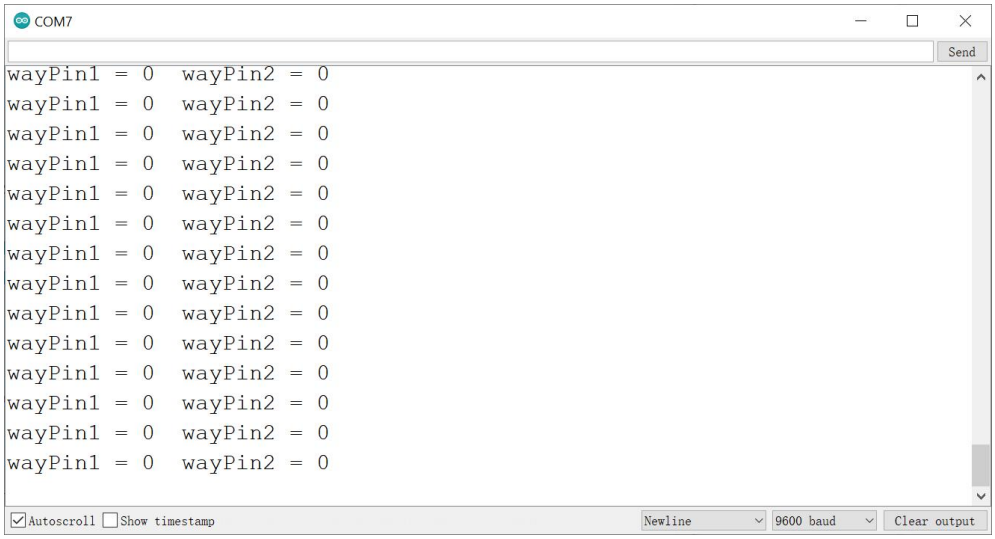
8.1 Read the value of tracking sensor

(1) Example code

```
#define wayPin1 9
#define wayPin2 10

boolean val1 = 0;
boolean val2 = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(wayPin1, INPUT);
  pinMode(wayPin2, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  val1 = digitalRead(wayPin1);
  val2 = digitalRead(wayPin2);
  Serial.print("wayPin1 = ");
  Serial.print(val1);
  Serial.print("    ");
  Serial.print("wayPin2 = ");
  Serial.println(val2);
  delay(100);
}
```

(2) Experimental phenomenon

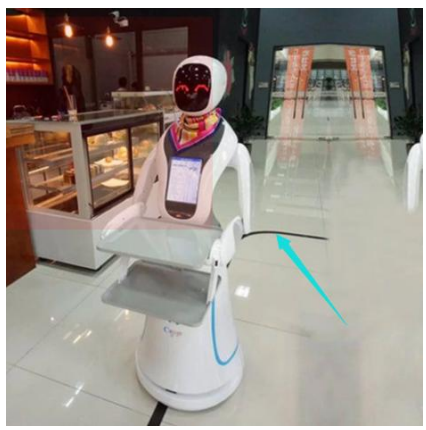
Open the serial port monitor of the Arduino IDE, and you can see that the values measured by the tracking sensor are printed. It is placed on the bright table, with a value of 0. Raise the little snail to a certain height, or place it on a black object. The value is 1.



8.2 Realization of the tracking function

We provide a sheet of paper printed with black lines. The little snail follows the black line.

There are some restaurant robots on the market now. If there are black tracks on the ground, they are basically realized by using tracking sensors. For example, the following figure.



Principle of tracking:

Value on the left	Value on the right	Direction
1	1	Forward
1	0	Turn left
0	1	Turn right
0	0	Stop

(1) Example code

```
#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to Control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to Control the speed of right motor

#define wayPin1 9
#define wayPin2 10
boolean L_tval = 0;
boolean R_tval = 0;

void setup() {
  Serial.begin(9600);     //Set the baud rate of serial port to 9600
  pinMode(INA, OUTPUT);    //Set the pin of controlling the motor to output
  pinMode(ENA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(wayPin1, INPUT);
  pinMode(wayPin2, INPUT);
}

void loop() {
  car_tracking();
}

void car_tracking()
{
  L_tval = digitalRead(wayPin1);
  R_tval = digitalRead(wayPin2);
  Serial.print(L_tval);
  Serial.print("    ");
  Serial.println(R_tval);
  delay(100);
  if((L_tval == 1) && (R_tval == 1))
  {
    car_forward();
  }
  else if((L_tval == 1) && (R_tval == 0))
  {
    car_left();
```

```
    }
    else if((L_tval == 0) && (R_tval == 1))
    {
        car_right();
    }
    else if((L_tval == 0) && (R_tval == 0))
    {
        car_stop();
    }
}

//The little snail moves forward
void car_forward()
{
    digitalWrite(INA, HIGH);    //Output high level and control the left motor forward
    analogWrite(ENA, 50);       //PWM output to control the speed of left motor
    digitalWrite(INB, HIGH);    //Output high level and control the right motor forward
    analogWrite(ENB, 50);       //PWM output to control the speed of right motor
}

//The little snail turns left
void car_left()
{
    digitalWrite(INA, LOW);
    analogWrite(ENA, 70);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 80);
}
//The little snail turns right
void car_right()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 80);
    digitalWrite(INB, LOW);
    analogWrite(ENB, 70);
}
//Stop
void car_stop()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 0);
    digitalWrite(INB, HIGH);
```

```
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

The little snail follows the black line.

Tip: If the little snail will run out of the black line and stop, modify it yourself to reduce the forward speed and left or right speed in the code, and slowly try to adjust.

8.3 Circle the little snail

There are some robot competitions. A circle of black tracks will be drawn on the periphery of the competition field, the robots inside can't run out. A simple way is to use tracking sensors and the robots return when the black lines are detected. It is also the function of the sweeping robot to prevent falling down the stairs. The sweeping robot is equipped with a tracking sensor (infrared diode) at the bottom. When the distance is relatively large, the infrared reflection is weak, indicating that it is a place in the air. Then let the robot return, and it will not fall down the stairs.

This experiment needs a larger field. We have provided large drawings. If you want to do this experiment, you can use black tape to stick a large circle.

(1) Example code

```
/*
  * create by straysnail
  * 2022/3/20
  */
#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to Control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to Control the speed of right motor

#define wayPin1 9
#define wayPin2 10
boolean L_tval = 0;
boolean R_tval = 0;
```

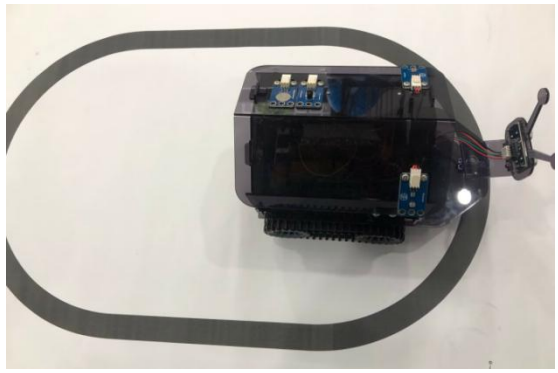
```
void setup() {
  Serial.begin(9600);     //Set the baud rate of serial port to 9600
  pinMode(INA, OUTPUT);    //Set the pin of controlling the motor to output
  pinMode(ENA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(wayPin1, INPUT);
  pinMode(wayPin2, INPUT);
}

void loop() {
  car_circle();
}

void car_circle()
{
  L_tval = digitalRead(wayPin1);
  R_tval = digitalRead(wayPin2);
  Serial.print(L_tval);
  Serial.print("    ");
  Serial.println(R_tval);
  delay(100);
  if((L_tval == 1) && (R_tval == 1))
  {
    car_back();
    delay(100);
    car_left();
    delay(200);
  }
  else if((L_tval == 1) && (R_tval == 0))
  {
    car_right();
    delay(100);
  }
  else if((L_tval == 0) && (R_tval == 1))
  {
    car_left();
    delay(100);
  }
  else if((L_tval == 0) && (R_tval == 0))
  {
```

```
      car_forward();
    }
}

//The little snail moves forward
void car_forward()
{
   digitalWrite(INA, HIGH);    //Output high level and control the left motor forward
   analogWrite(ENA, 50);       //PWM output to control the speed of left motor
   digitalWrite(INB, HIGH);    //Output high level and control the right motor forward
   analogWrite(ENB, 50);       //PWM output to control the speed of right motor
}

void car_back()
{
   digitalWrite(INA, LOW);
   analogWrite(ENA, 50);
   digitalWrite(INB, LOW);
   analogWrite(ENB, 50);
}


//The little snail turns left
void car_left()
{
   digitalWrite(INA, LOW);
   analogWrite(ENA, 70);
   digitalWrite(INB, HIGH);
   analogWrite(ENB, 80);
}
//The little snail turns right
void car_right()
{
   digitalWrite(INA, HIGH);
   analogWrite(ENA, 80);
   digitalWrite(INB, LOW);
   analogWrite(ENB, 70);
}
//Stop
void car_stop()
{
   digitalWrite(INA, HIGH);
```

```
  analogWrite(ENA, 0);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

We provide a drawing. The little snail walks in the black circle, and can't walk out of it.



9.The little snail follows the light

Story: The little snail has been walking for most of the day. It is already dark, and the lighthouse in front is emitting white light. The little snail continues to move towards the lighthouse.

Photosensitive sensor: The photosensitive sensors installed on both sides of the little snail roof are analog sensors, which are mainly made according to the characteristics of the photosensitive resistor. The resistance of the photosensitive resistor will change with the change of the light intensity, that is, within a certain range, the stronger the light, the greater the measured value.

The IO port of the left photosensitive sensor	A1
The IO port of the right photosensitive sensor	A2

9.1 Read the values of the photosensitive sensors

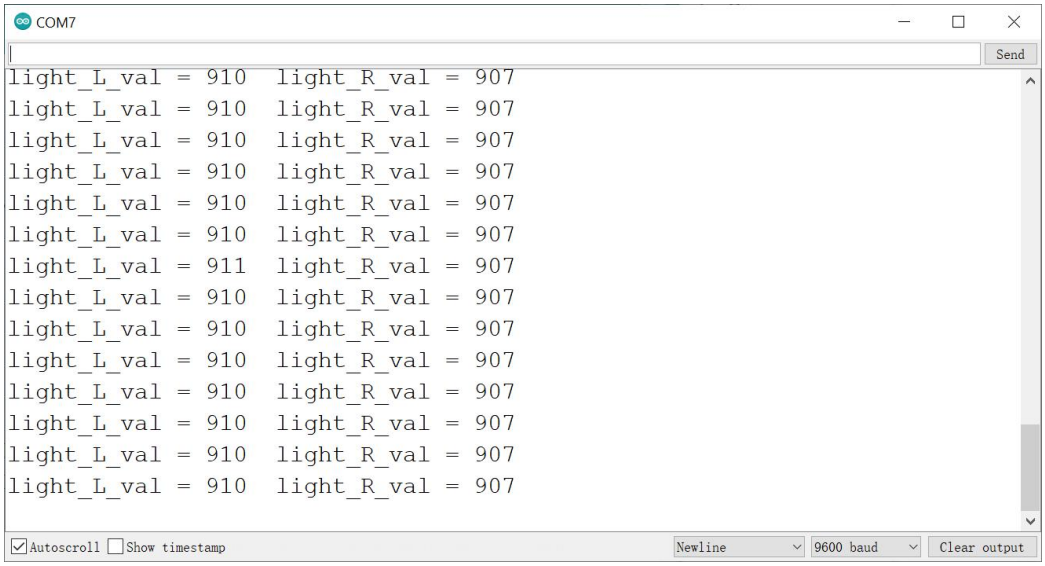
(1) Example code

```
/*
```

```
 * create by straysnail
 * 2022/2/28
 */
#define lightPin_L A1    //Define the pin of photosensitive sensor as A1
#define lightPin_R A2

int light_L_val;
int light_R_val;

void setup() {
  Serial.begin(9600);
  pinMode(lightPin_L, INPUT);    //Set to input
  pinMode(lightPin_R, INPUT);
}

void loop() {
  light_L_val = analogRead(lightPin_L);    //Read the analog value
  light_R_val = analogRead(lightPin_R);
  Serial.print("light_L_val = ");
  Serial.print(light_L_val);
  Serial.print("    ");
  Serial.print("light_R_val = ");
  Serial.println(light_R_val);
  delay(100);
}
```

(2) Experimental phenomenon

The serial port monitor prints the measured values of the two photosensitive sensors.



9.2 Light control lamp

In some residential areas, the lights will be turned on automatically at night. Many of them are realized by using photosensitive sensors. As long as the measured light is less than a certain value, the lights will be turned on automatically. We use the photosensitive sensor on the left side and the LED light of the little snail to achieve this function.

(1) Example code

```
#define lightPin_L A1
#define LEDPin A3

int light_L_val;

void setup() {
  Serial.begin(9600);
  pinMode(lightPin_L, INPUT);
  pinMode(LEDPin, OUTPUT);
}
```

```
void loop() {
  light_L_val = analogRead(lightPin_L);
  Serial.print("light_L_val = ");
  Serial.println(light_L_val);
  delay(100);
  if(light_L_val < 500)    //When it is lower than the set value, the LED will be on
  {
    digitalWrite(LEDPin, HIGH);
  }
  else
  {
    digitalWrite(LEDPin, LOW);
  }
}
```

(2) Experimental phenomenon

When the left photosensitive sensor is covered by hand or in a dark place, the LED light will light up.

9.3 The little snail follows the light

If two photosensitive sensors are placed on both sides, the three directions of a light source, which can be defined as front, left and right, can be determined according to the values measured by the two photosensitive sensors. When the three directions can be judged, the snail following the light can be realized.

Left photoresistor	Right photoresistor	Direction of travel
The measured value is greater than the set value	The measured value is greater than the set value	forward
The measured value is greater than the set value	The measured value is less than the set value	Turn left
The measured value is less than the set value	The measured value is greater than the set value	Turn right

(1) Example code

```
/*
 * create by straysnail
 * 2022/2/28
 */
```

```
#define L_light A1
#define R_light A2
int L_lightVal;
int R_lightVal;

//Define the pin of motors
#define INA 7
#define ENA 6
#define INB 8
#define ENB 5

void setup() {
  Serial.begin(9600);
  //Set the pin of the photosensitive sensors to input
  pinMode(L_light, INPUT);
  pinMode(R_light, INPUT);

  //Set the pin of the motors to output
  pinMode(INA, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(ENB, OUTPUT);
}

void loop() {
  L_lightVal = analogRead(L_light);
  R_lightVal = analogRead(R_light);
  Serial.print(L_lightVal);
  Serial.print("    ");
  Serial.println(R_lightVal);
  delay(100);
  if((L_lightVal >= 800) && (R_lightVal >= 800))
  {
    car_forward();
  }
  else if((L_lightVal >= 800) && (R_lightVal < 800))
  {
    car_left();
  }
  else if((L_lightVal < 800) && (R_lightVal >= 800))
  {
```

```
        car_right();
    }
    else if((L_lightVal < 800) && (R_lightVal < 800))
    {
        car_stop();
    }
}

//The little snail moves forward
void car_forward()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 155);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 155);
}

//The little snail turns left
void car_left()
{
    digitalWrite(INA, LOW);
    analogWrite(ENA, 150);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 150);
}

//The little snail turns right
void car_right()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 150);
    digitalWrite(INB, LOW);
    analogWrite(ENB, 150);
}

//Stop
void car_stop()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 0);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 0);
```

```
}
```

(2) Experimental phenomenon

Turn on the flashlight of a mobile phone and shine on the photosensitive sensors on the little snail. When the two photosensitive sensors are exposed at the same time, the small snail moves forward. When only the left side is illuminated, it turns left, and when only the right side is illuminated, turns right.

Tip: If the flashlight shines on the photosensitive sensors of the little snail, it does not move very much, then you need to adjust the size of the setting value "if(L_lightVal>=800)&&(R_lightVal>=800)", and slowly adjust it to be appropriate one.

10.Autonomous obstacle avoidance and following of the little snail

Story: On the way to the lighthouse, the little snail finds that there are many creatures going to the lighthouse on the road ahead, and it follows them slowly along the way.

Ultrasonic module: The ultrasonic module is almost necessary for the maker robots, simple and practical, whose shape is like eyes. The eyes of the little snail use ultrasonic modules.

This ultrasonic module is used to detect distance. Two eyes, one is to transmit ultrasonic waves of a certain frequency, and the other is to receive the reflected ultrasonic waves. The speed of sound wave is mainly related to the propagation medium and temperature. We often use 340m/s, which is the speed of sound wave propagating in standard air at 15 ℃. We also need to know the time from sending sound waves to receiving sound waves to calculate the distance of the object in front. The calculation formula is: distance=340m/s * t/2.

Little knowledge: The sound frequency heard by human ears is within a range. When the sound frequency is higher than the human hearing range, it belongs to ultrasonic wave.

Note: The ultrasonic module we use is based on IIC communication protocol. There are many products on the market using two common IO ports to control ultrasound. The IIC protocol is two-wire system, including the signal lines SDA and SCL, which are corresponding to the A4 and A5 pins of the Arduino UNO main board. They can be connected to multiple sensor modules of IIC protocol at the same time. Our ultrasonic and OLED screens are both IIC protocols, which means that two precious IO ports can be saved.

Method of controlling IIC Ultrasonic:

IIC address: 0X57

Command format:

address	command	Return value	explain

Write address : 0XAE	0X01		Start ranging
Read address : 0XAF		BYTE_H BYTE_M BYTE_L	The output distance is ((BYTE_H<<16) +(BYTE_M<<8)+BYTE_L)/1000 Unit : mm

Write 0X01 to the module, and the module starts ranging, and wait for more than 100mS (maximum ranging time of the module). Read the 3 distance values directly, which are "BYTE_ H", "BYTE_ M" and "BYTE_ L". The method of distance calculation is as follows (unit: mm):

Distance=((BYTE_H<<16)+(BYTE_M<<8)+BYTE_ L)/1000

Arduino IIC communication protocol knowledge link:

https://www.arduino.cc/en/reference/wire

Combined with the Example code provided by us, it will be better understood.

The IO port of Ultrasonic Trig pin	A4
The IO port of Ultrasonic Echo pin	A5

10.1 Read the distance measured by the ultrasonic module

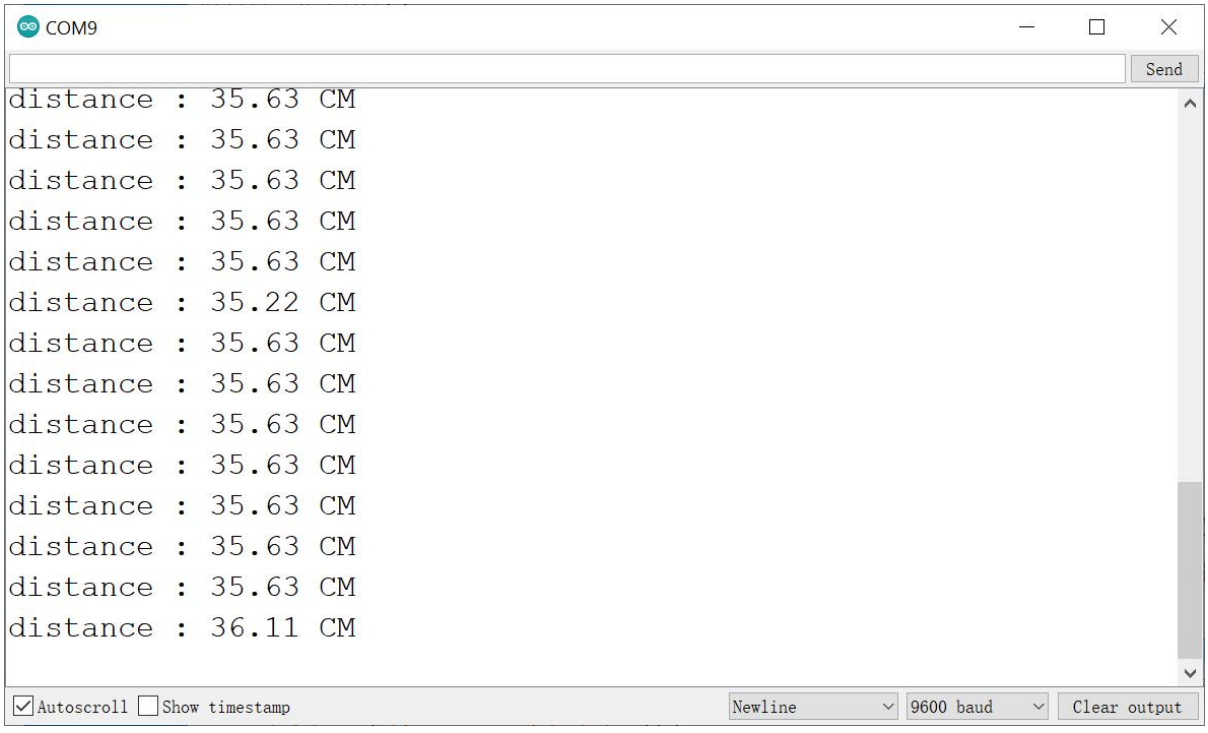
(1) Example code

```
#include <Wire.h>
float distance = 0;    //Decimal distance value
float ds[3];    //Create an array of float numbers to store the values read

void setup()
{
  Serial.begin(9600);    //Define the baud rate of serial port to 9600, which is the factory default value
  Wire.begin();
  Serial.println("Ultrasonic-IIC start ranging：");
}
```

```
void loop()
{
    char i = 0;
    ds[0]=0;
    ds[1]=0;
    ds[2]=0;
    Wire.beginTransmission(0x57);     //Initial Address is 0X57
    Wire.write(1);                    //Writing command 0X01: 0X01 is the command to start measuring
    Wire.endTransmission();
    delay(120);                       //Measure cycle delay: One cycle is 100ms, set 120ms to keep some
allowance
    Wire.requestFrom(0x57,3);         //The address is 0X57, and read 3 8-bit distance values
    while (Wire.available())
    {
       ds[i++] = Wire.read();
    }
    distance=(ds[0]*65536+ds[1]*256+ds[2])/10000; //Calculated as cm value
    Serial.print("distance : ");
    if ((1<=distance)&&(600>=distance))              //Value between 1cm-6m
    {
      Serial.print(distance);
       Serial.print(" CM ");
    }
    else
    {
      Serial.print(" - - - - ");                    //Invalid value is displayed as- - - -
    }
    Serial.println();
    delay(100);                                     //Adjust the measure cycle
}
```

(2) Experimental phenomenon

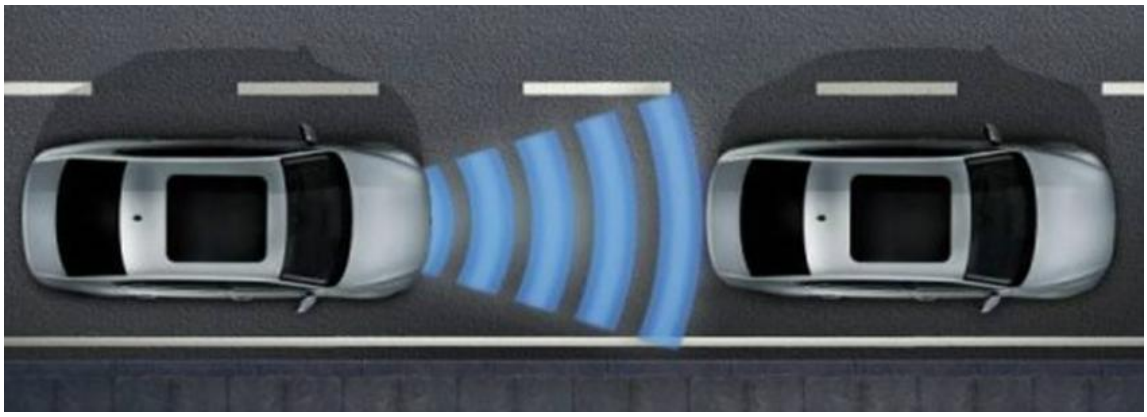
The serial port monitor prints the distance value measured by the ultrasonic module. Put your hand in front of the ultrasonic probe, and you can see the measured distance between the ultrasonic module and the hand is relatively accurate.



Tip: IIC communication protocol is very common. So please try to Learn more: https://www.arduino.cc/en/reference/wire

10.2 Following function of the little snail

In recent years, with the rapid development of artificial intelligence, automatic driving has been pushed onto the tide. Automatic car following is an essential function of automatic driving. To achieve car following, we need to know the distance from the vehicle in front. Of course, the ranging equipment mentioned here is not ultrasonic ranging, but high-end laser radar and depth camera. We use ultrasonic module to achieve simple simulation.



Straight line car following principle is as shown in the following table:

Distance measured by ultrasonic wave (cm)	The movement of the snail

15<Distance<30	forward
8<Distance<=15	stop
Distance<=8	back off
Distance>=30	stop

(1) Example code

```
#include <Servo.h>    //Import the library files of steering gear
Servo head_servo;    //Create a steering gear instance to control a steering gear
#include <Wire.h>
float distance = 0;    //Decimal distance value
float ds[3];    //Create an array of float numbers to store the values read

#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to control the speed of right motor

void setup() {
  Serial.begin(9600);
  Wire.begin();
  pinMode(INA, OUTPUT);    //Set the pins controlling the motors to output
  pinMode(ENA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(ENB, OUTPUT);
  head_servo.attach(12,500,2500);    //Set the steering gear pin as A2, and the range is 500-2500ms
}

void loop() {
  head_servo.write(90);    //90degrees
  distance = checkdistance(); //Get the value returned by ultrasonic function
  if(distance < 8)
  {
    car_back();
  }
  else if((distance >= 8) && (distance <= 16))
  {
    car_stop();
  }
  else if((distance > 16) && (distance <= 25))
```

```
    {
      car_forward();
    }
    else
    {
      car_stop();
    }
}

//Ultrasonic ranging function
float checkdistance() {
    char i = 0;
    ds[0]=0;
    ds[1]=0;
    ds[2]=0;
    Wire.beginTransmission(0x57);      //Initial Address is 0X57
    Wire.write(1);                     //Writing command 0X01: 0X01 is the command to start measuring
    Wire.endTransmission();
    delay(100);                        //Measure cycle delay: One cycle is 100ms, set 120ms to keep some
allowance
    Wire.requestFrom(0x57,3);          //The address is 0X57, and read 3 8-bit distance values
    while (Wire.available())
    {
     ds[i++] = Wire.read();
    }
    distance=(ds[0]*65536+ds[1]*256+ds[2])/10000; //Calculated as cm value
    Serial.print("distance : ");
    if ((1<=distance)&&(600>=distance))                  //Value between 1cm-6m
    {
    Serial.print(distance);
     Serial.print(" CM ");
    }
    else
    {
    Serial.print(" - - - - ");                          //Invalid value is displayed as- - - -
    }
    Serial.println();
    delay(10);                                          //Adjust the measure cycle
    return distance;
}

//The little snail moves forward
```

```
void car_forward()
{
  digitalWrite(INA, HIGH);   //Output high level and control the left motor forward
  analogWrite(ENA, 100);      //PWM output to control the speed of left motor
  digitalWrite(INB, HIGH);   //Output high level and control the right motor forward
  analogWrite(ENB, 100);      //PWM output to control the speed of right motor
}
//The little snail moves backward
void car_back()
{
  digitalWrite(INA, LOW);   //Output low level and control the left motor backward
  analogWrite(ENA, 100);     //PWM output to control the speed of left motor
  digitalWrite(INB, LOW);   //Output low level and control the right motor backward
  analogWrite(ENB, 100);     //PWM output to control the speed of right motor
}
//Stop
void car_stop()
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 0);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

Use your palm or other objects to block the front of the little snail, about 10cm away. The little snail will not move. Then slowly move the hand forward, and the little snail will follow. If the hand slowly approaches the front of the little snail, it will move back.

Tip: It is recommended to adjust the distance and speed set in the program to achieve your satisfaction.

10.3 Automatic obstacle avoidance function

Automatic obstacle avoidance is also an essential function of automatic driving. When an obstacle is detected in front of the car, the laser radar and other sensors should also detect the surrounding conditions, and then choose the method to avoid the obstacle. Of course, we are just achieve the simplest ultrasonic automatic obstacle avoidance function.

When the front distance measured by ultrasonic wave is >15cm: the steering gear of the head will be kept at 90 degrees, and the little snail moves forward.

When the front distance measured by ultrasonic wave is <=15cm: the little snail stops and the steering gear rotates to 180 degrees, that is, the ultrasonic module is put to the left to measure the distance on the left. Then it will be swung to the right and measure the distance on the right. By comparing the distance between the left and right sides, the little snail turns to the side with a larger distance and continues to move forward.

(1) Example code

```
#include <Servo.h>    //Import the library files of steering gear
Servo head_servo;    //Create a steering gear instance to control a steering gear
#include <Wire.h>
float distance = 0;    //Decimal distance value
float ds[3];    //Create an array of float numbers to store the values read
int distance1; //Variable 1 for storing ranging values
int distance2; //Variable 2 for storing ranging values
int distance3; //Variable 3 for storing ranging values

#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to control the speed of right motor

void setup() {
  Serial.begin(9600);
  Wire.begin();
  pinMode(INA, OUTPUT);    //Set the pins controlling the motors to output
  pinMode(ENA, OUTPUT);
  pinMode(INB, OUTPUT);
  pinMode(ENB, OUTPUT);
  head_servo.attach(12,500,2500);    //Set the steering gear pin as A2, and the range is 500-2500ms
  head_servo.write(90);    //90 degrees
  delay(200);
}
```

```
void loop() {
  func_avoid(); // The sub function of obstacle avoidance
}

void func_avoid()
{
  distance1 = checkdistance(); //Get the value returned by ultrasonic function
  Serial.print("distance1 = ");
  Serial.println(distance1);    //Print the values detected by ultrasonic sensor with new line
  if(distance1 > 15)
  {
    car_forward();
  }
  else
  {
    car_stop();
    head_servo.write(180);    //The head of the snail turns left
    delay(300);
    distance2 = checkdistance(); //Get the left distance value
    delay(100);
    head_servo.write(10);     //The head of the snail turns right
    delay(500);
    distance3 = checkdistance(); //Get the left distance value
    delay(100);
    if(distance2 > distance3)
    {
      car_left();    //Turn left
      head_servo.write(90);
      delay(300);    //The delay time determines the rotation angle
    }
    else
    {
      car_right();
      head_servo.write(90);
      delay(300);
    }
  }
}

//Ultrasonic ranging function
float checkdistance() {
  char i = 0;
```

```
  ds[0]=0;
  ds[1]=0;
  ds[2]=0;
  Wire.beginTransmission(0x57);     //Initial Address is 0X57
  Wire.write(1);                    //Writing command 0X01: 0X01 is the command to start measuring
  Wire.endTransmission();
  delay(100);                       //Measure cycle delay: One cycle is 100ms, set 120ms to keep some
allowance
  Wire.requestFrom(0x57,3);         //The address is 0X57, and read 3 8-bit distance values
  while (Wire.available())
  {
    ds[i++] = Wire.read();
  }
  distance=(ds[0]*65536+ds[1]*256+ds[2])/10000; //Calculated as cm value
  Serial.print("distance : ");
  if ((1<=distance)&&(600>=distance))              //Value between 1cm-6m
  {
  Serial.print(distance);
    Serial.print(" CM ");
  }
  else
  {
  Serial.print(" - - - - ");                      //Invalid value is displayed as- - - -
  }
  Serial.println();
  delay(10);                                      //Adjust the measure cycle
  return distance;
}

//The little snail moves forward
void car_forward()
{
  digitalWrite(INA, HIGH);   //Output high level and control the left motor forward
  analogWrite(ENA, 100);      //PWM output to control the speed of left motor
  digitalWrite(INB, HIGH);   //Output high level and control the right motor forward
  analogWrite(ENB, 100);      //PWM output to control the speed of right motor
}
//The little snail moves backward
void car_back()
{
  digitalWrite(INA, LOW);    //Output low level and control the left motor backward
  analogWrite(ENA, 100);     //PWM output to control the speed of left motor
```

```
  digitalWrite(INB, LOW);    //Output low level and control the right motor backward
  analogWrite(ENB, 100);     //PWM output to control the speed of right motor
}
//The little snail turns left
void car_left()
{
  digitalWrite(INA, LOW);
  analogWrite(ENA, 200);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 200);
}
//The little snail turns right
void car_right()
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 200);
  digitalWrite(INB, LOW);
  analogWrite(ENB, 200);
}
//Stop
void car_stop()
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 0);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

The little snail moves forward, and it stops when detecting an obstacle in front. It turns its head to the left (measure the left distance, when there is an obstacle on the left), and then turns to the right (measure the right distance). The right distance is greater than the left distance, so the little snail turns right.

11.OLED ulstrasonic range finder

Ultrasonic range finder requires ultrasonic and display screen. A 0.96-inch blue OLED display screen is installed on the little snail.

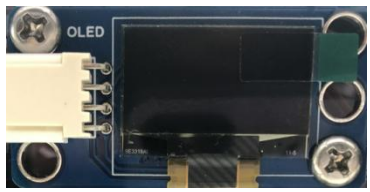
OLED display screen: Display screen is the most commonly used module for the frequently-used devices for modern people, such as mobile phones, computers, televisions,

etc. An electronic device will look backward without a display screen. The display screen really makes the interaction between electronic devices and people intuitive, simple and practical.

The IO port for the OLED display SDA pin	A4
The IO port for the OLED display SCL pin	A5

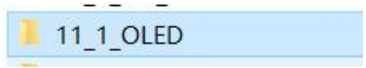
11.1 OLED display screen

Although the OLED screen is small, there are many functions that can be realized. Now let's learn the basic functions of controlling the OLED screen.



(1) Example code

Because the code is so long, please open the example code to view.



(2) Experimental phenomenon

You can see that the OLED display screen displays various patterns, lines and characters.

Tip: Be familiar with function statements that control OLED display characters and patterns.

11.2 Ultrasonic range finder

Sometimes it is not easy to measure something with a ruler, so you can choose to use an ultrasonic range finder, which is very convenient to use. As long as it is aligned, you can know the relatively accurate distance value.

(1) Example code

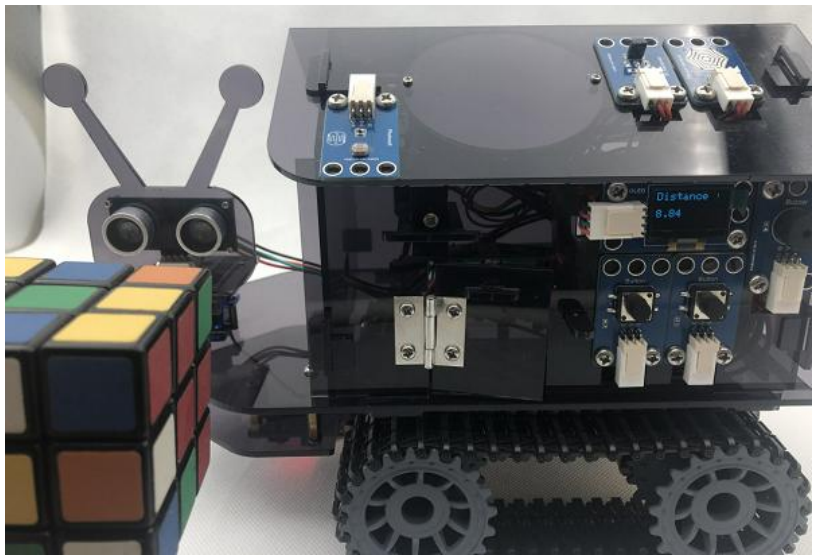
```
#include <Wire.h>
float distance = 0;  //Decimal distance value
float ds[3];  //Create an array of float numbers to store the values read
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define btnPin1 2  //Define button pin
//The variable is used to receive the value detected by the button
boolean btnVal1;
int count;
boolean ranging_state = 0;  //Used to decide whether to start ranging

void setup() {
  Serial.begin(9600);
  Wire.begin();
  pinMode(btnPin1,INPUT);  //Set pin 11 to output

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

   // Clear the buffer
  display.clearDisplay();
  display.display();
```

```
}

void loop() {
  btnVal1 = digitalRead(btnPin1);  //Read the button value and assign it to btnVal1
  if(btnVal1 == 0)  //If button 1 is pressed
  {
    delay(10);
    if(btnVal1 == 0)
    {
      boolean i = 1;
      while(i == 1)
      {
        btnVal1 = digitalRead(btnPin1);  //Read the button value and assign it to btnVal1
        if(btnVal1 == 1)
        {
          count++;
          Serial.println(count);
          i = 0;
        }
      }
      ranging_state = count % 2; //Calculate the remainder. If it is singular, the value is
equal to 1, and if it is even, the value is equal to 0
    }

  }
  if(ranging_state == 1)
  {
    distance = checkdistance();  //Read the value detected by ultrasonic sensor
    display.setTextSize(2);             // Normal 1:1 pixel scale
    display.setTextColor(SSD1306_WHITE);        // Draw white text
    display.setCursor(0,0);             // Start at top-left corner
    display.println(F("Distance : "));
    display.println(distance);
    display.display();
    display.clearDisplay();
  }
  else
  {
    display.setTextSize(2);             // Normal 1:1 pixel scale
    display.setTextColor(SSD1306_WHITE);        // Draw white text
    display.setCursor(0,0);             // Start at top-left corner
    display.println(F("Distance : "));
```

```
    display.println(distance);
    display.display();
    display.clearDisplay();
  }
}

//Ultrasonic ranging function
float checkdistance() {
  char i = 0;
  ds[0]=0;
  ds[1]=0;
  ds[2]=0;
  Wire.beginTransmission(0x57);    //Initial Address is 0X57
  Wire.write(1);                   //Writing command 0X01: 0X01 is the command to start
measuring
  Wire.endTransmission();
  delay(120);                      //Measure cycle delay: One cycle is 100ms, set 120ms to keep
some allowance
  Wire.requestFrom(0x57,3);        //The address is 0X57, and read 3 8-bit distance values
  while (Wire.available())
  {
   ds[i++] = Wire.read();
  }
  distance=(ds[0]*65536+ds[1]*256+ds[2])/10000; //Calculated as cm value
  Serial.print("distance : ");
  if ((1<=distance)&&(600>=distance))           //Value between 1cm-6m
  {
  Serial.print(distance);
   Serial.print(" CM ");
  }
  else
  {
  Serial.print(" - - - - ");                   //Invalid value is displayed as- - - -
  }
  Serial.println();
  delay(10);                                   //Adjust the measure cycle
  return distance;
}
```

(2) Experimental phenomenon

The ultrasonic module is aimed at the direction of distance measurement. Press button 1, and the OLED display screen will display the distance value measured by the ultrasonic wave. Press button 1 again to stop distance measurement.



11.3 OLED displays picture

As a screen, it should be able to display pictures. Now let's learn how to display pictures with OLED.

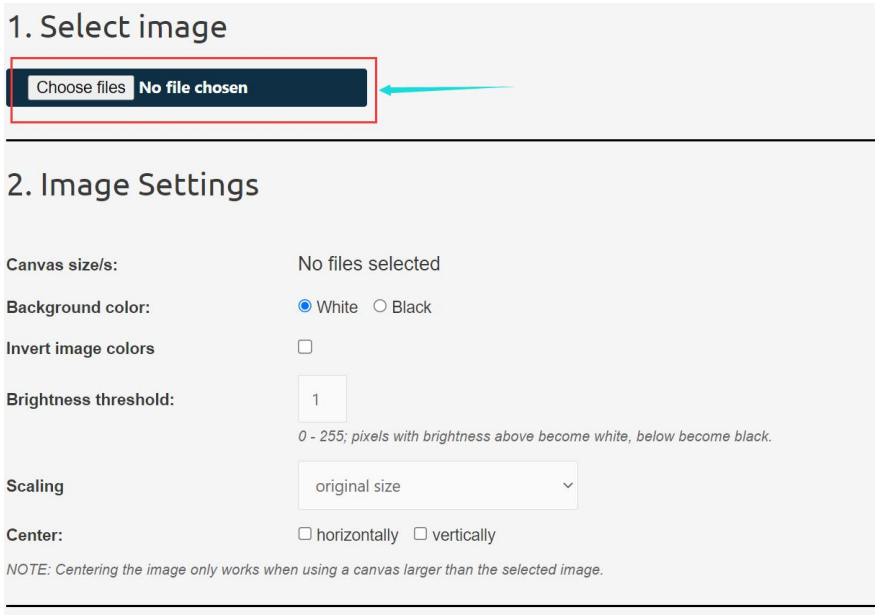
(1) Get your pictures ready

The pixels of our OLED are 128 * 64P, so we need to prepare 128 * 64P pictures. We have provided sample pictures in the “4. sample picture” folder.

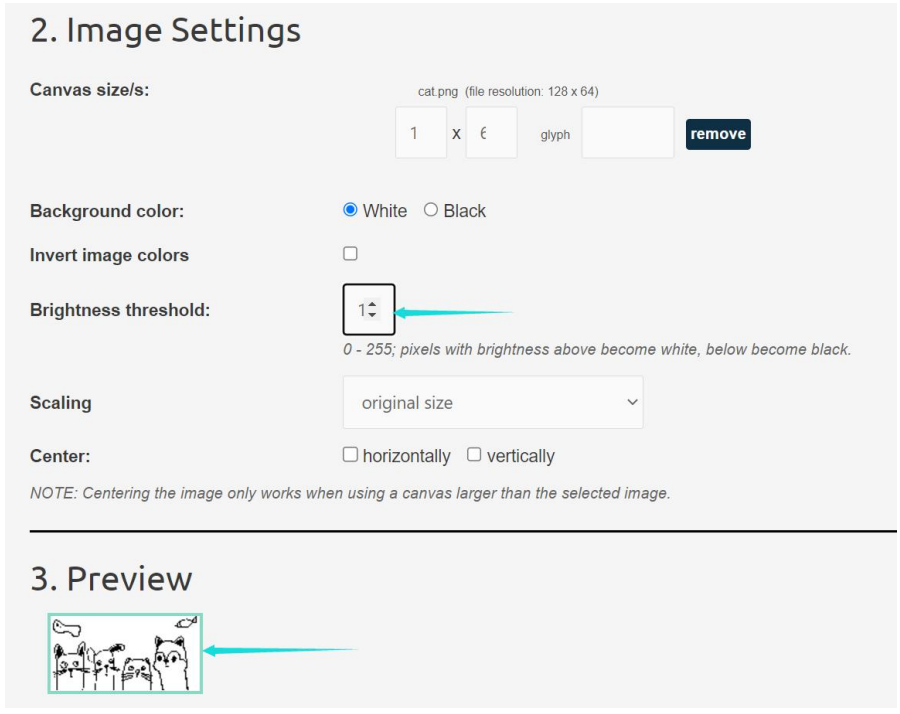
(2) Convert the picture to hexadecimal

Link for picture to hexadecimal: https://diyusthad.com/image2cpp

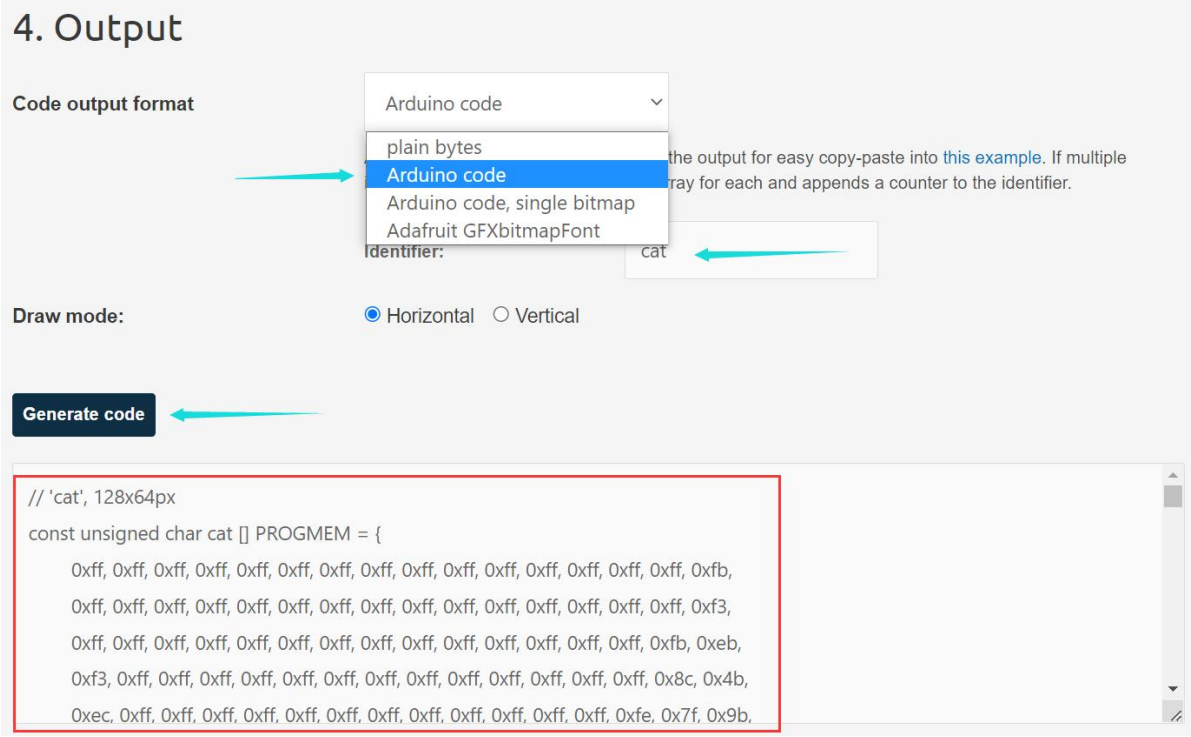
Open the link, and drag the “cat.png” sample image to the “Select file”.



When you just bring in the picture, you can see that the effect picture in “3. Preview” is not very clear. You can click the up arrow of Brightness threshold to adjust it until the picture becomes clear, as shown below.

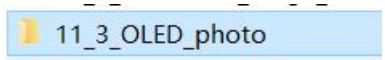


Select “Arduino code”, rename it “cat”, and then click “Generate code” to generate the hexadecimal code for Arduino C. Copy and paste the hexadecimal code into the example code provided by us.



(3) Example code

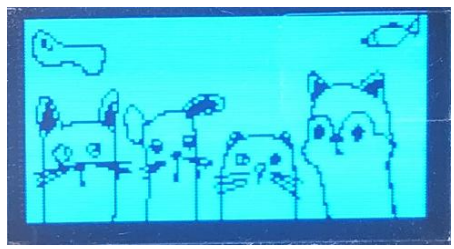
Because the code is so long, please open the example code to view.



If you convert other pictures, remember to modify the name of the corresponding hexadecimal code in the code, such as “cat display. drawBitmap(0, 0, cat, 128, 64, 1);”.

(4) Experimental phenomenon

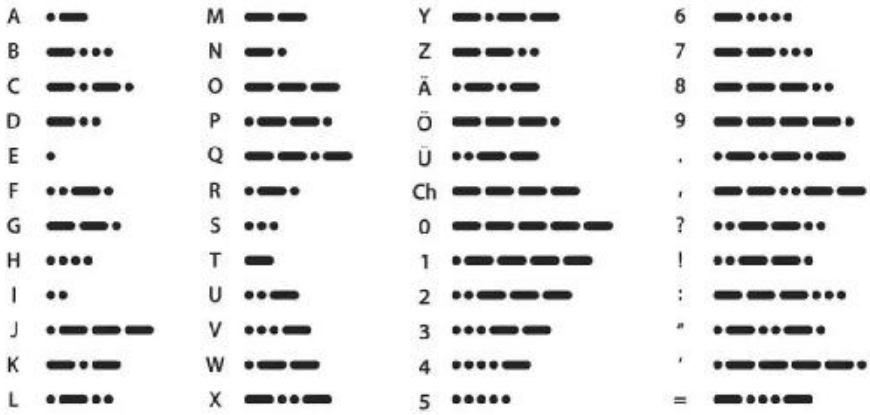
The OLED screen displays the picture successfully.



12.Morse code gate

Many people should have seen some stories in the movie that use Morse code to communicate, such as sending Morse code to each other by tapping, using a flashlight, etc.

Little knowledge: Morse code is a kind of signal code that goes on and off, expressing different English letters, numbers and punctuation marks through different permutations. Morse code consists of two basic signals: short dot signal " •", read "Di", and long signal "—" for a certain time, read "Da". Interval time: Di=1t, Da=3t, Di Da interval=1t, character interval=3t, word interval=7t.



Morse code

12.1 OneButton

The "OneButton" library file allows you to use various ways of buttons, including double click, long press and other functions. See the link for details: https://github.com/mathertel/OneButton

(1) Example code

```
#include "OneButton.h"


// Setup a new OneButton on pin 2.
OneButton button1(2, true);
// Setup a new OneButton on pin 13.
OneButton button2(13, true);


void setup() {
  Serial.begin(9600);
  // link the button 1 functions.
  button1.attachClick(click1);
  button1.attachDoubleClick(doubleclick1);
  button1.attachLongPressStart(longPressStart1);
  button1.attachLongPressStop(longPressStop1);
```

```
  button1.attachDuringLongPress(longPress1);

  // link the button 2 functions.
  button2.attachClick(click2);
  button2.attachDoubleClick(doubleclick2);
  button2.attachLongPressStart(longPressStart2);
  button2.attachLongPressStop(longPressStop2);
  button2.attachDuringLongPress(longPress2);
}

void loop() {
  // keep watching the push buttons:
  button1.tick();
  button2.tick();
}

// ----- button 1 callback functions

// This function will be called when the button1 was pressed 1 time (and no 2. button press followed).
void click1() {
  Serial.println("Button 1 click.");
} // click1


// This function will be called when the button1 was pressed 2 times in a short timeframe.
void doubleclick1() {
  Serial.println("Button 1 doubleclick.");
} // doubleclick1


// This function will be called once, when the button1 is pressed for a long time.
void longPressStart1() {
  Serial.println("Button 1 longPress start");
} // longPressStart1


// This function will be called often, while the button1 is pressed for a long time.
void longPress1() {
  Serial.println("Button 1 longPress...");
} // longPress1
```

```
// This function will be called once, when the button1 is released after beeing pressed for a long time.
void longPressStop1() {
  Serial.println("Button 1 longPress stop");
} // longPressStop1


// ... and the same for button 2:

void click2() {
  Serial.println("Button 2 click.");
} // click2


void doubleclick2() {
  Serial.println("Button 2 doubleclick.");
} // doubleclick2


void longPressStart2() {
  Serial.println("Button 2 longPress start");
} // longPressStart2


void longPress2() {
  Serial.println("Button 2 longPress...");
} // longPress2

void longPressStop2() {
  Serial.println("Button 2 longPress stop");
} // longPressStop2
```

(2) Experimental phenomenon

Open the serial port monitor, click the button, double click the button, and long press the button to see the corresponding string printed.

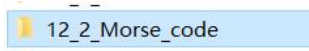
12.2 Morse code gate

Open the door by entering the Morse code. Morse code 'F' is used as the password in the code.

F ••—•

(1) Example code

Because the code is so long, please open the example code to view.



(2) Experimental phenomenon

Enter the Morse password by short press and long press of button 1. The OLED screen displays the entered password. The Morse code for opening the door is "F", that is, " • •— •". If the input is correct, the OLED displays success and the door will open; If it is an error input, the OLED displays error. Wait for 1 second, and then enter the password again when the OLED displays again.

13.Infrared remote control the little snail

Infrared remote control is a very useful wireless communication method. It is used in most electrical equipment at home, such as TV, air conditioner, fan, etc. As long as you press the button of infrared remote control, the appliance will turn on the corresponding function, which is very convenient to use.

The IO port of main board or expansion board to connect	D3

13.1 Read the received value

The infrared remote control and infrared receiving module are used together. The infrared remote control provided by us has many buttons, so what is the value of each button? This lesson will print out the values of infrared remote control received by the infrared receiving module through the serial port monitor.

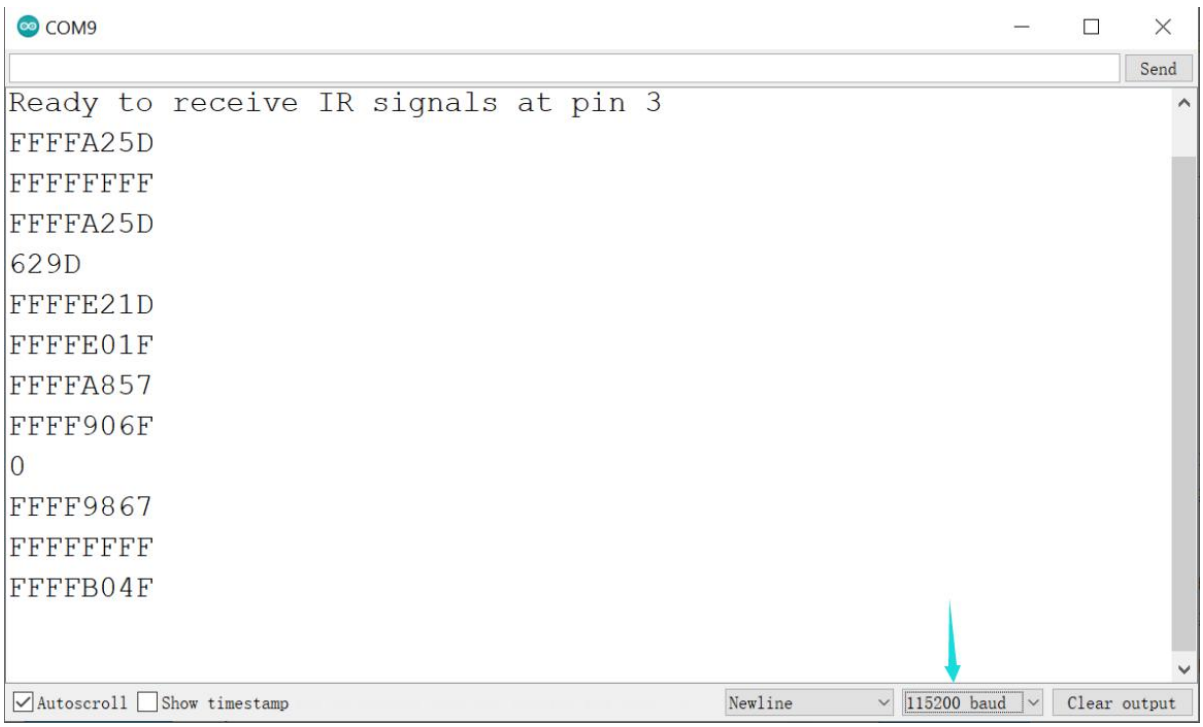
(1) Example code

```
#include <IRremote.h>
#define IR_RECEIVE_PIN    3    //Define the pin of infrared receiving module
IRrecv IrReceiver(IR_RECEIVE_PIN);
decode_results results;
int ir_val;

void setup() {
   Serial.begin(115200);    //Define the baud rate to 115200
   Serial.println("Enabling IRin");
   IrReceiver.enableIRIn();    // Start the receiver
   IrReceiver.blink13(true); // Enable feedback LED

   Serial.print(F("Ready to receive IR signals at pin "));
   Serial.println(IR_RECEIVE_PIN);
}

void loop() {
   if (IrReceiver.decode(&results))
   {
```

```
        ir_val = results.value;
        //IrReceiver.printResultShort(&Serial);
        Serial.println(ir_val,HEX);

        IrReceiver.resume(); // Receive the next value
    }
    delay(100);
}
```

(2) Experimental phenomenon

Open the serial port monitor, press the key with the infrared remote controller toward the infrared receiving module, and you can see that the serial port monitor prints the received value.



Tip: If there is no response, it is recommended to check whether there is a battery in the infrared remote controller.

13.2 Infrared remote control the little snail

Many toy cars use infrared remote control to control the movement of the car. Wireless remote control is very interesting.

The infrared remote controller provided by us has four direction keys. We use these four direction keys and OK keys to control the movement of the little snail.

(1) Example code

```
#include <IRremote.h>
#define IR_RECEIVE_PIN    3    //Define the pin of infrared receiving module
IRrecv IrReceiver(IR_RECEIVE_PIN);
decode_results results;
int ir_val;
int ir_FFFF;
int ir_val_data;
#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to control the speed of right motor

void setup() {
   Serial.begin(115200);    //Define the baud rate to 115200
   Serial.println("Enabling IRin");
   IrReceiver.enableIRIn();    // Start the receiver
   //IrReceiver.blink13(true); // Enable feedback LED

   Serial.print(F("Ready to receive IR signals at pin "));
   Serial.println(IR_RECEIVE_PIN);
   pinMode(INA, OUTPUT);    //Set the pins controlling the motors to output
   pinMode(ENA, OUTPUT);
   pinMode(INB, OUTPUT);
   pinMode(ENB, OUTPUT);
}

void loop() {
   if(IrReceiver.decode(&results))
   {
      ir_val = results.value;
      //IrReceiver.printResultShort(&Serial);
      Serial.println(ir_val,HEX);
      if(ir_val == 0xFFFF)
      {
         ir_FFFF = ir_val;
      }
      else
      {
         ir_val_data = ir_val;
      }
```

```
        switch(ir_val_data)
        {
          case 0x18E7: car_forward(); break;    //The little moves forward
          case 0x4AB5: car_back(); break;         //The little moves backward
          case 0x10EF: car_left(); break;         //The little turns left
          case 0x5AA5: car_right(); break;        //The little turns right
        }
        IrReceiver.resume(); // Receive the next value
    }
    else
    {
        car_stop(); //Stop
    }
    delay(100);
}

//The little moves forward
void car_forward()
{
    digitalWrite(INA, HIGH);    //Output high level and control the left motor forward
    analogWrite(ENA, 200);       //PMW output to control the speed of left motor
    digitalWrite(INB, HIGH);    //Output high level and control the right motor forward
    analogWrite(ENB, 200);       //PMW output to control the speed of right motor
}
//The little snail moves backward
void car_back()
{
    digitalWrite(INA, LOW);    //Output low level and control the left motor backward
    analogWrite(ENA, 200);      //PMW output to control the speed of left motor
    digitalWrite(INB, LOW);    //Output low level and control the right motor backward
    analogWrite(ENB, 200);      //PMW output to control the speed of right motor
}
//The little snail turns left
void car_left()
{
    digitalWrite(INA, LOW);
    analogWrite(ENA, 200);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 200);
}
//The little snail turns right
void car_right()
```

```
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 200);
  digitalWrite(INB, LOW);
  analogWrite(ENB, 200);
}
//Stop
void car_stop()
{
  digitalWrite(INA, HIGH);
  analogWrite(ENA, 0);
  digitalWrite(INB, HIGH);
  analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

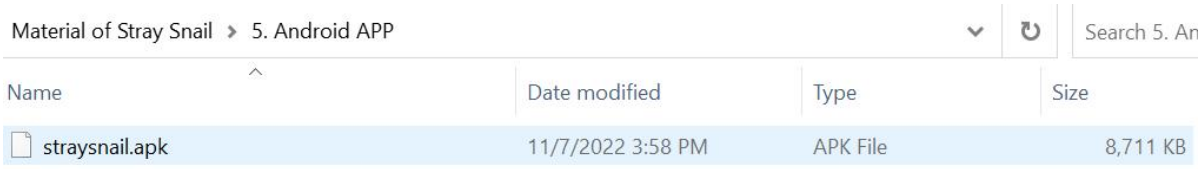
Press and hold the infrared remote control direction button, and the little snail will run in the corresponding direction. After loosening, the little snail will stop.

14.Bluetooth control the snail

Mobile phones are essential for modern young people. Bluetooth is an essential function of mobile phones, so we have designed and developed a Bluetooth APP to specifically control various functions of the little snail.

Download Android APP: Search “straysnail” in Google play

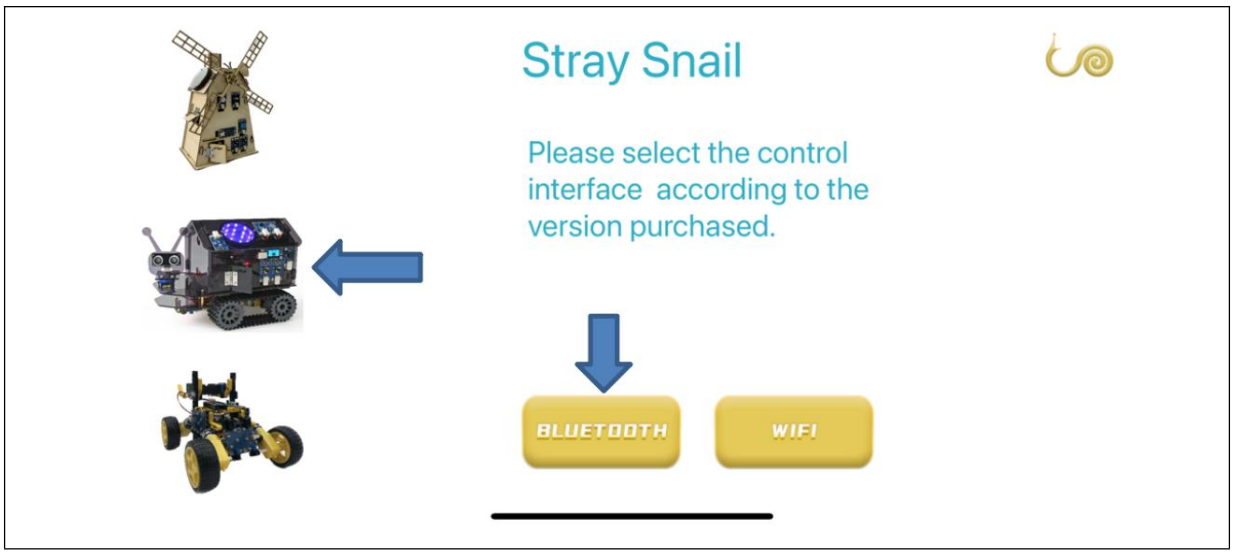
Or use the installation package provided by us:



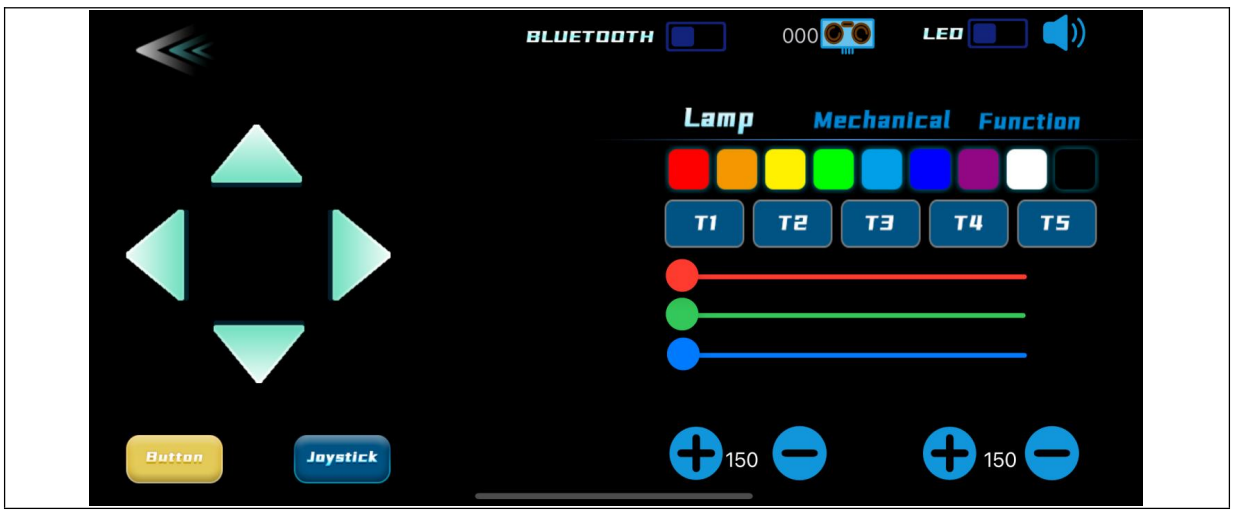
Download iOS APP: Search “straysnail” in APP Store

APP interface introduction:

Select the Stray Snail in the following picture on the APP interface, and then select the Bluetooth control mode, as shown below.



Enter the control interface, as shown below.



14.1 Read the values of Bluetooth APP

The Bluetooth APP also controls the snail by sending characters or digital commands to the snail's Bluetooth module. What characters are the buttons of our app? In order to intuitively see what characters are received, we print the received characters on the serial port monitor.

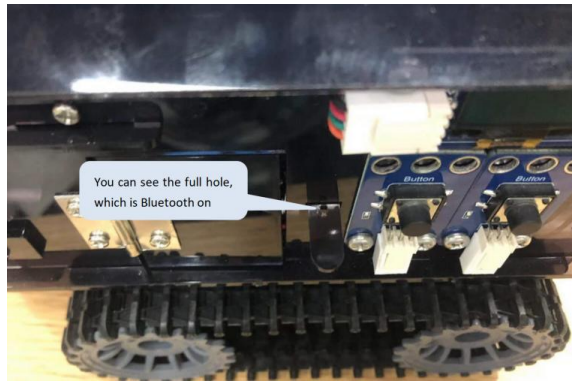
(1) Example code

```
char bleStr = "";

void setup() {
  Serial.begin(9600);

}

void loop() {
  while(Serial.available() > 0)  //Judge whether the serial port has received the value
  {
    bleStr = Serial.read();  //Read the value received by serial port
    Serial.println(bleStr);
    delay(10);
  }
}
```

(2) Experimental phenomenon

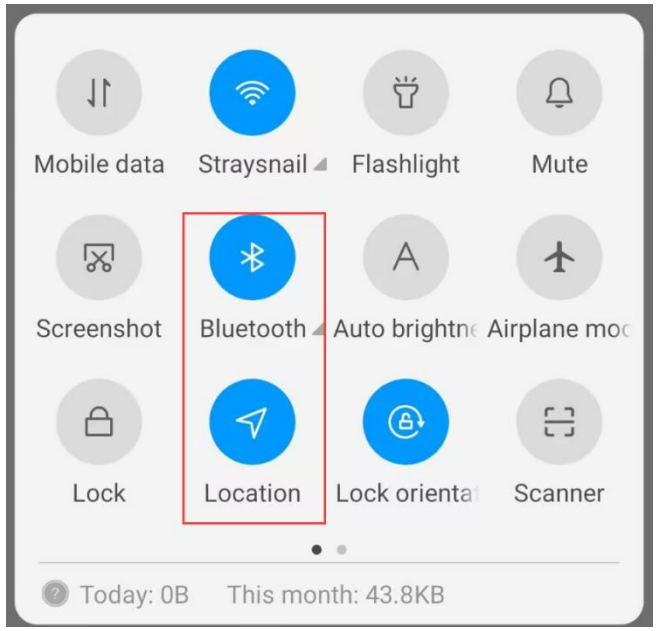
Turn on the serial port monitor, turn on the Bluetooth switch of the little snail (that is, pull the switch on the right side of the door), turn on the mobile phone APP, connect the Bluetooth, click the button on the interface, and you can see that the serial port monitor prints the received characters.

1. **Turn on the Bluetooth switch of the snail after burning the code** (be careful not to turn it on first and then burn the code, which will result in code burning failure).



2. Open the APP for Bluetooth search and connection

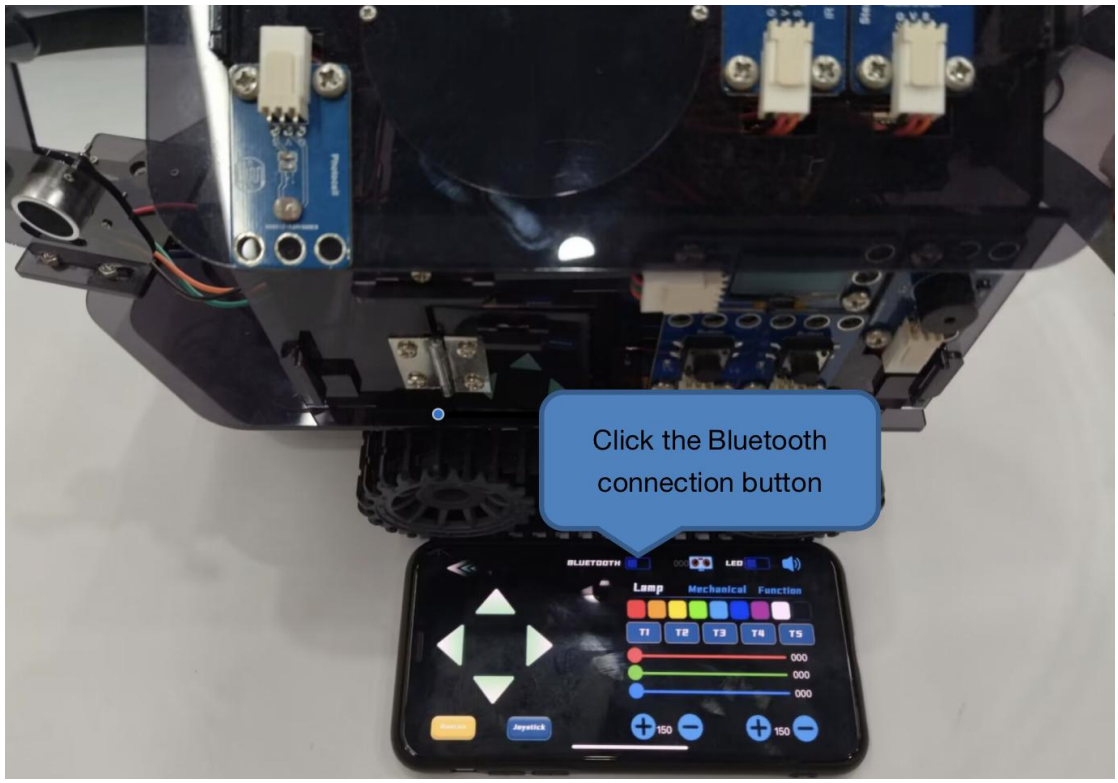
First,If you have an Android phone, you need to open the Bluetooth and location functions of your phone. iOS requires Bluetooth to be enabled



There are two ways to connect the Bluetooth:

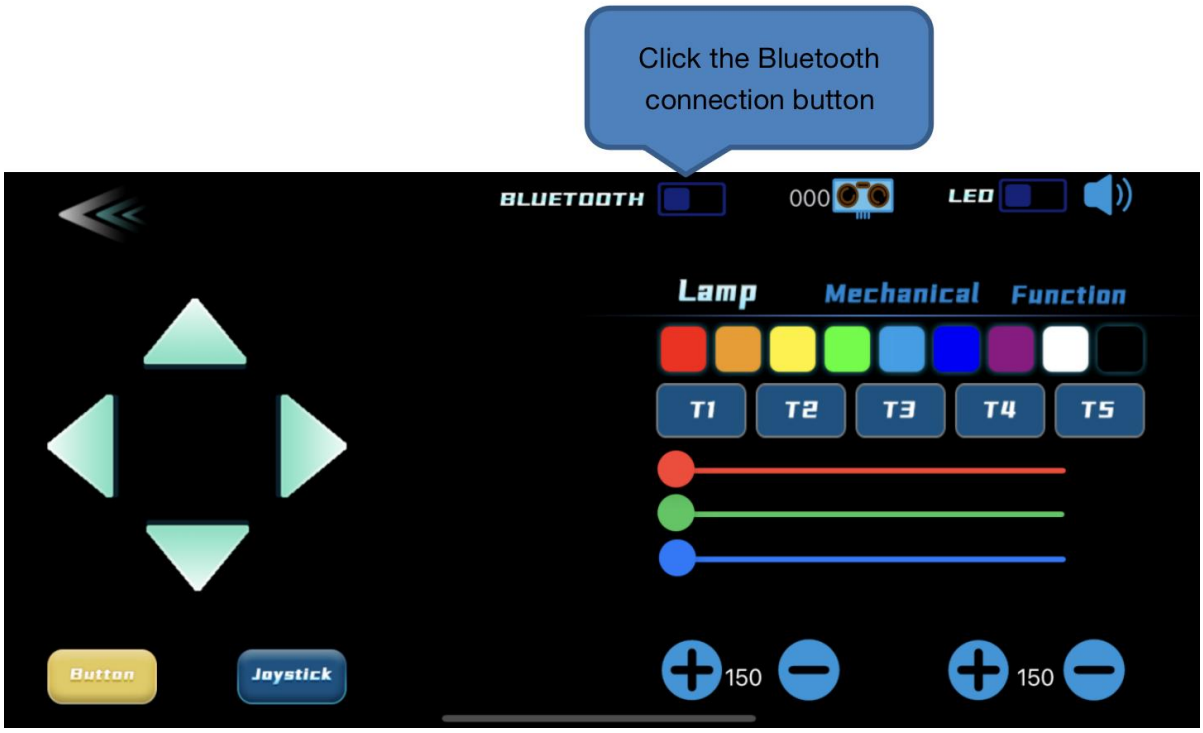
（1）Close to connect

Turn on the power of Stray Snail, and turn on the APP on your mobile phone. Get close to the snail, and click the Bluetooth button on the interface. Bluetooth can automatically connect. If the connection is successful, a pop-up window will appear on the APP interface indicating “Stray Snail: Bluetooth has automatically connected successfully”, as shown below.

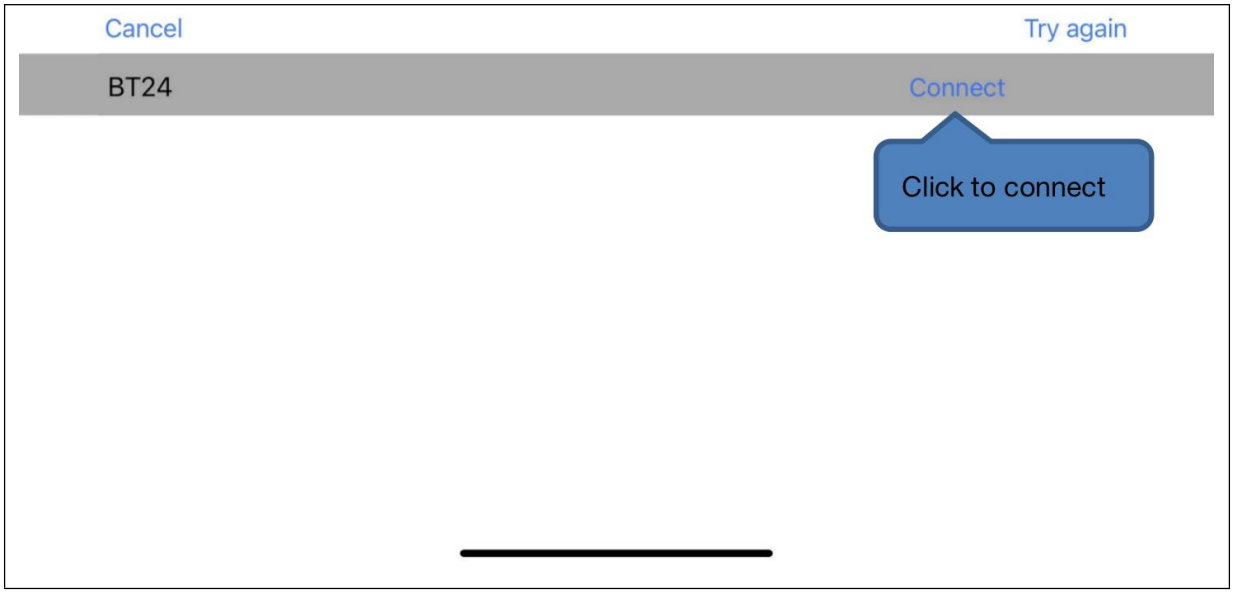


（2）Connect manually

Open the APP interface and click the Bluetooth button, as shown below.

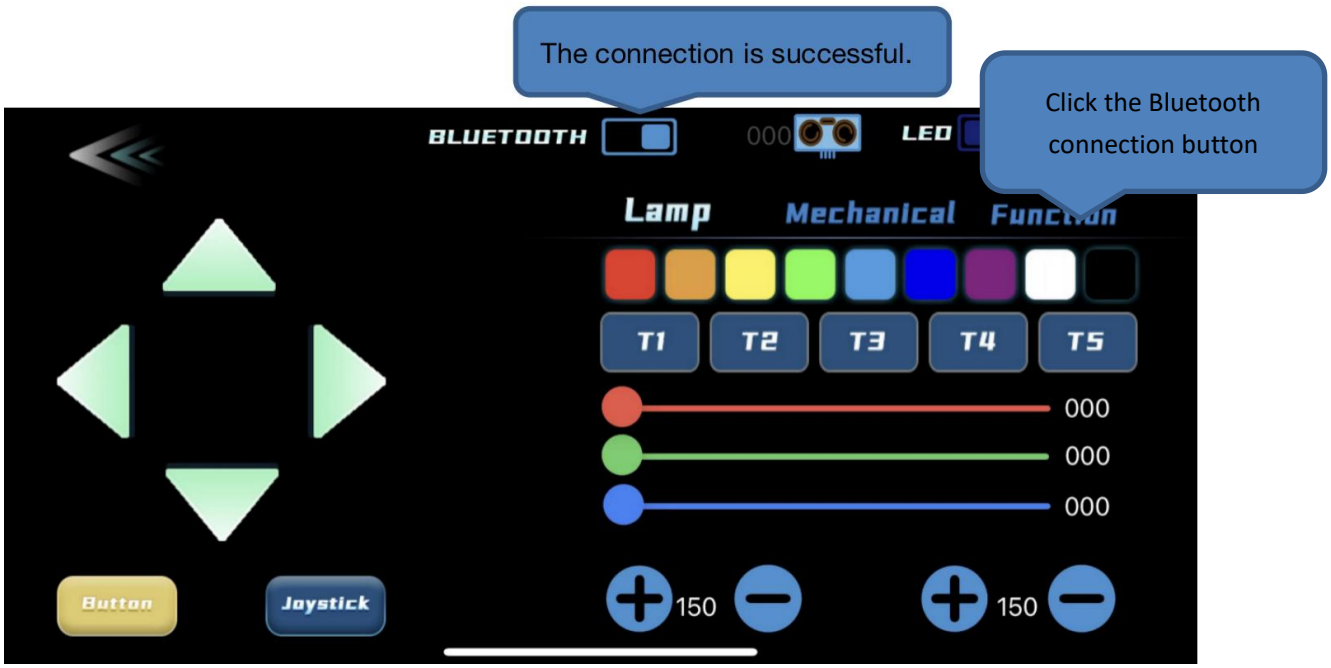


Bluetooth name is BT24 or JDY-23.



（3）Send an order

After the connection is successful, you can click the button on the interface to print the received value in the serial port monitor of the arduino IDE.



Open the serial port monitor of the Arduino IDE to see the received Bluetooth values.

14.2 Bluetooth APP controls the movement of the snail

In the last lesson, we know the button values of the Bluetooth APP, so we can use the direction buttons on the APP interface to control the movement of the little snail.

(1) Example code

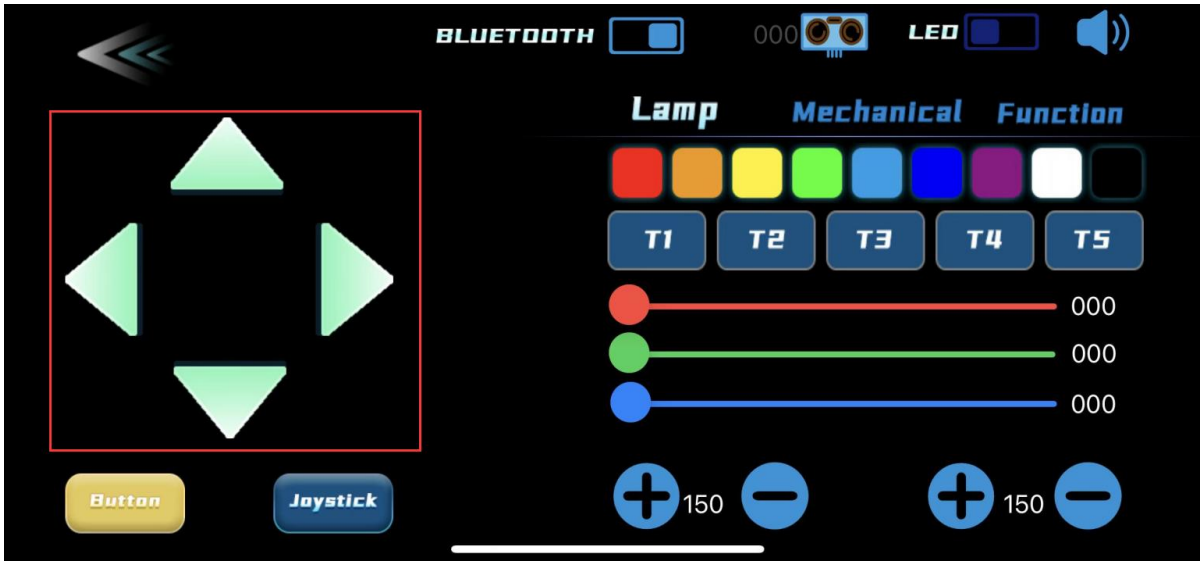
```
/*
  * create by straysnail
  * 2022/3/6
  */
char bleVal;

#define INA 7    //Define pin 7 to control the direction of left motor
#define ENA 6    //Define pin 6 to control the speed of left motor
#define INB 8    //Define pin 8 to control the direction of right motor
#define ENB 5    //Define pin 5 to control the speed of right motor

void setup() {
   Serial.begin(9600);
   pinMode(INA, OUTPUT);    //Set the pins controlling the motors to output
   pinMode(ENA, OUTPUT);
   pinMode(INB, OUTPUT);
   pinMode(ENB, OUTPUT);
}
```

```
void loop() {
  if(Serial.available() > 0)
  {
    String bleVal1 = Serial.readStringUntil('*');
    if(bleVal1[0] == '$')
    {
      bleVal = bleVal1[1];
      Serial.println(bleVal);
    }
    switch(bleVal)
    {
      case 'F': car_forward(); break;
      case 'B': car_back(); break;
      case 'L': car_left(); break;
      case 'R': car_right(); break;
      case 'S': car_stop(); break;
    }
  }
}

//The little snail moves forward
void car_forward()
{
  digitalWrite(INA, HIGH);   //Output high level and control the left motor forward
  analogWrite(ENA, 200);      //PWM output to control the speed of left motor
  digitalWrite(INB, HIGH);   //Output high level and control the right motor forward
  analogWrite(ENB, 200);      //PWM output to control the speed of right motor
}
//The little snail moves backward
void car_back()
{
  digitalWrite(INA, LOW);   //Output low level and control the left motor backward
  analogWrite(ENA, 200);     //PWM output to control the speed of left motor
  digitalWrite(INB, LOW);   //Output low level and control the right motor backward
  analogWrite(ENB, 200);     //PWM output to control the speed of right motor
}
//The little snail turns left
void car_left()
{
  digitalWrite(INA, LOW);
  analogWrite(ENA, 200);
  digitalWrite(INB, HIGH);
```

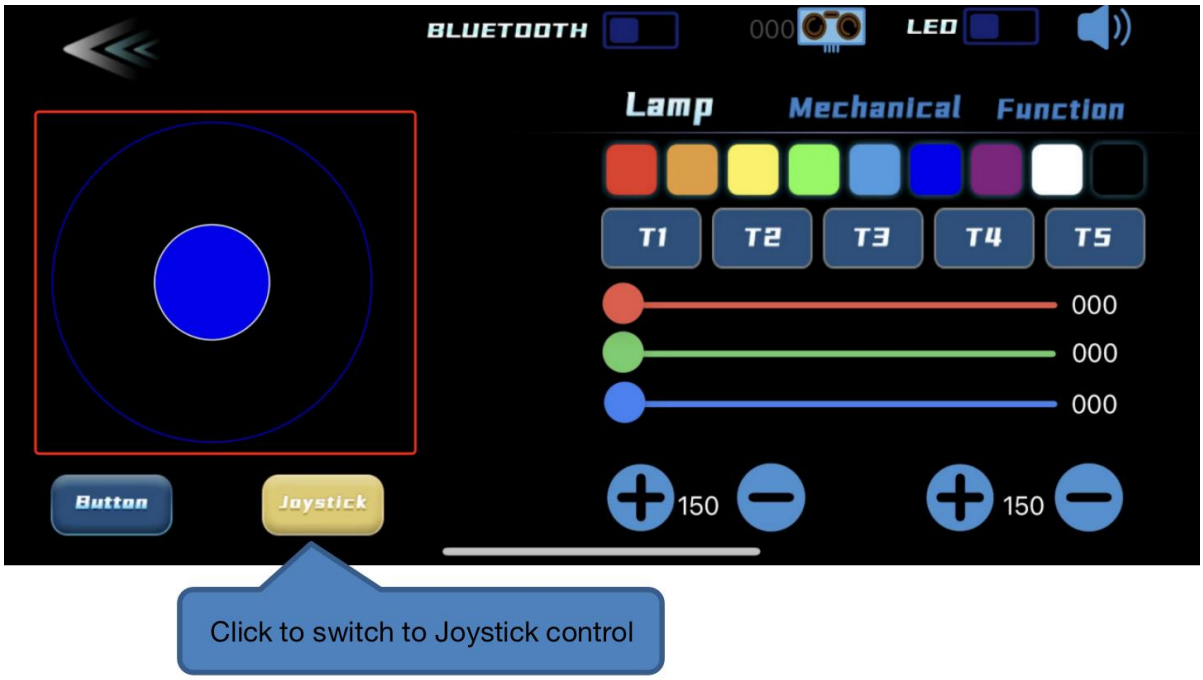
```
    analogWrite(ENB, 200);
}
//The little snail turns right
void car_right()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 200);
    digitalWrite(INB, LOW);
    analogWrite(ENB, 200);
}
//Stop
void car_stop()
{
    digitalWrite(INA, HIGH);
    analogWrite(ENA, 0);
    digitalWrite(INB, HIGH);
    analogWrite(ENB, 0);
}
```

(2) Experimental phenomenon

Turn on the Bluetooth switch of the little snail. After the mobile phone APP is successfully connected to the Bluetooth, press the direction button on the APP interface and then the little snail will drive in the corresponding direction.



You can also use a virtual joystick to control the car's movement. Click to switch to the joystick control interface, as shown below.



15.Infrared remote controls the snail

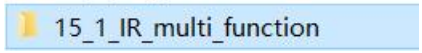
We have learned the function of infrared remote control before, but it is just a simple way to control the movement of the snail. Now we can control and switch various functions through infrared remote control.

15.1 Infrared remote controls multi-function little snail

There are many buttons for infrared remote control. We try to use all the buttons.

(1) Example code

Because the code is so long, please open the example code to view.



(2) Experimental phenomenon

The direction buttons of the infrared remote controller controls the movement direction of the little snail, button 1 controls the LED light, button 2 controls the buzzer, button 3 controls the tunnel light, button 4 and button 5 control the switch of the door, button 6 and button 7 control the switch of the window. Press button 8 to start the tracking function, press button 9 to start the following function, press button * to start the automatic obstacle avoidance function, press button 0 to start the light tracking function, and press button OK to exit all these function modes.

16.Press button to switch multiple functions

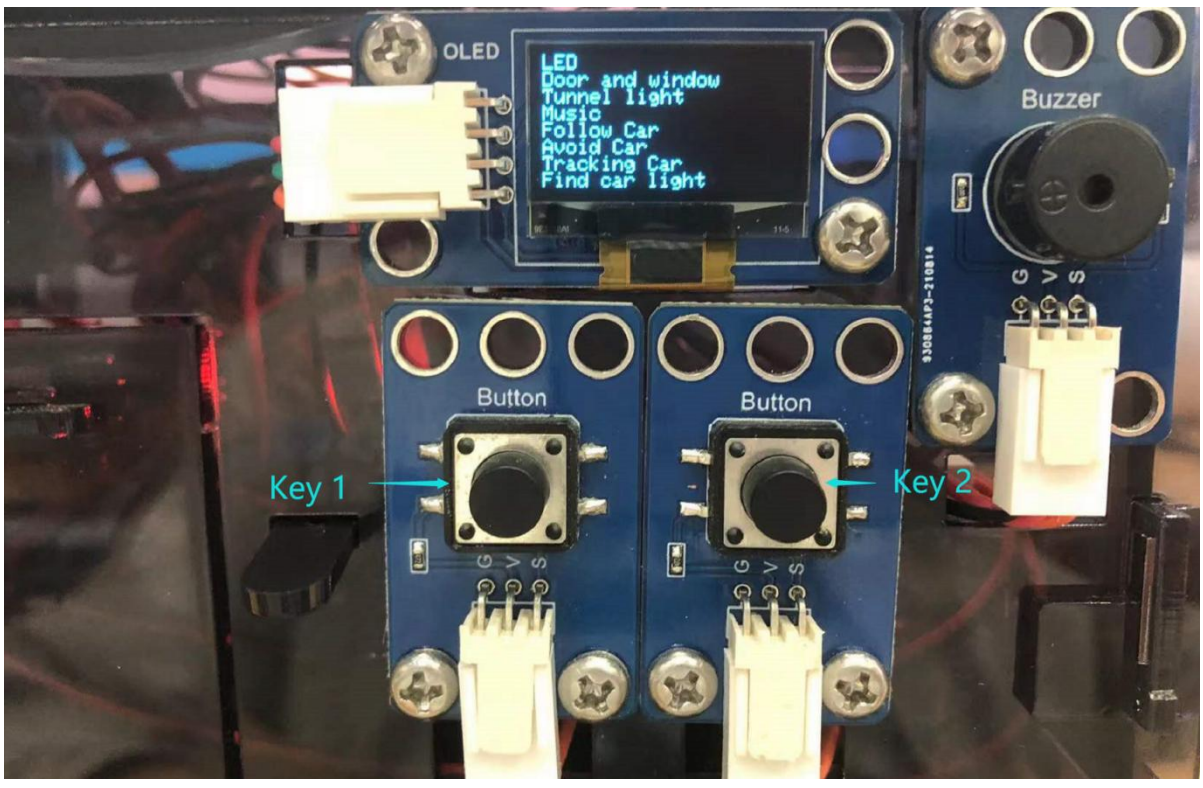
Think about it. There are two buttons and an OLED screen on the little snail. Isn't it possible to do simple interaction between the buttons and the screen? Therefore, we write the code that the OLED screen will display the names of the eight major functions of the little snail, and then press the button to select and determine the open functions. The code is a bit long, but it is not too complex. After the previous study, I believe you can understand it if you look carefully. The code in this lesson is also the code in our demo tutorial.

16.1 Button and srceen interactive selection function

(1) Example code

The code is long. Open the example code “16_ 1_ Button_ switch_ multifunction” provided by us by using the Arduino IDE.

(2)Experimental phenomenon and operation steps



Note: When you start the Follow Car, Avoid Car and Find car light functions, the little snail will move. Be careful not to let it fall off the table.

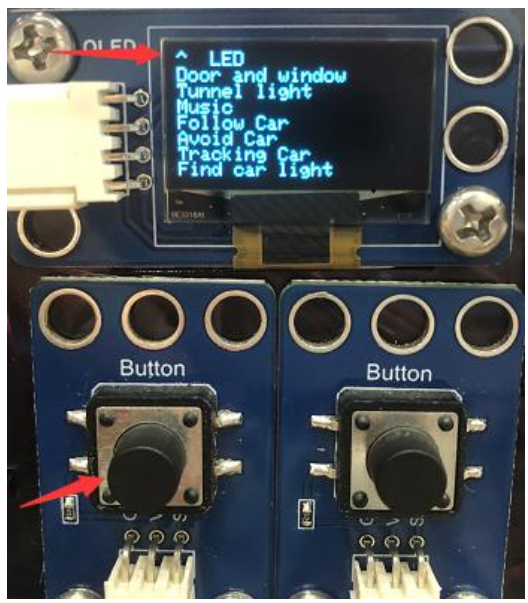
1. Easy trial steps

- The OLED screen displays the names of eight functions.
- Click button 1 slowly to select the function.
- Double click button 1 to enter the function, and the screen will display the operation method of the corresponding function.
- Press and hold button 1 and then press button 2 to exit the current function.
- You can also use Bluetooth APP to connect the Bluetooth module of Stray Snail to control the driving direction of it.

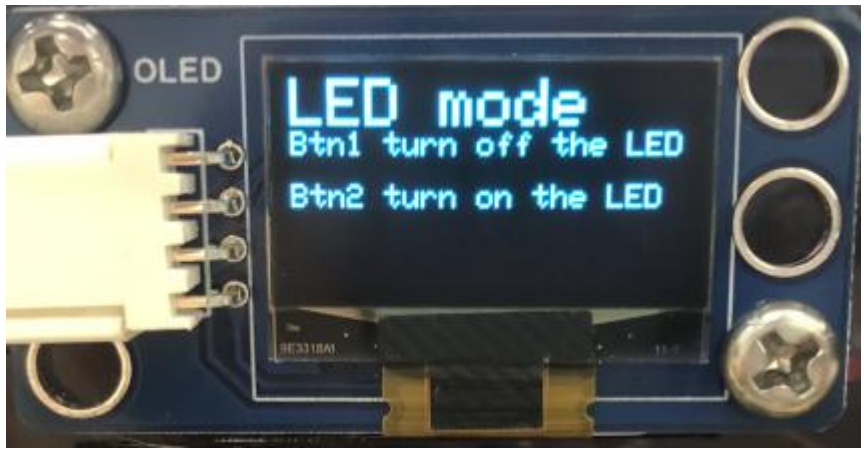
2. Detailed trial operation steps

2.1. LED

Click button 1 slowly to choose the LED function. The “^”in front of “LED”means that it is selected, as shown below.



Double click button 1 quickly to enter the function of controlling the LED. Click button 1 once, and LED comes on. Click button 1 again, and LED is off.

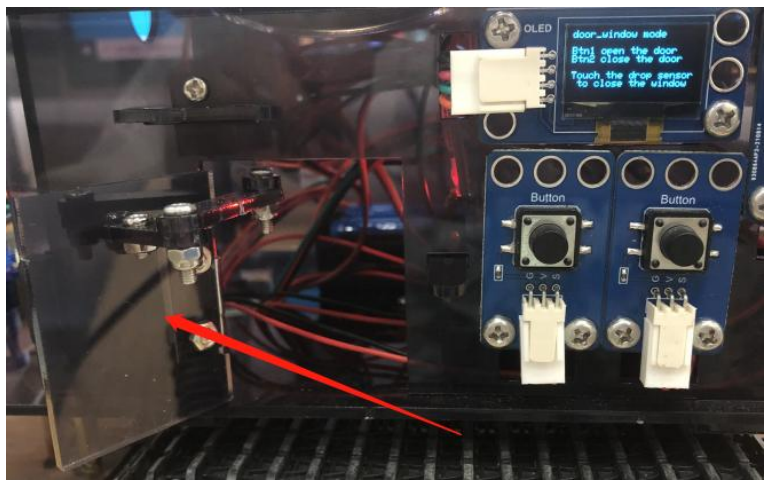


Press and hold button 1, and click button 2 immediately, the current function will be exited.

2.2. Door and window

Click button 1 slowly to choose Door and window function.

Double click button 1 quickly to enter the function of controlling door and window. Click button 2 once, and the door and window open. Click button 2 again, they close.

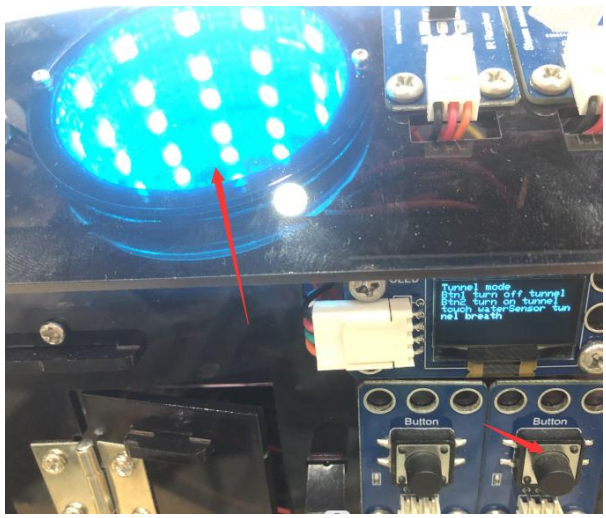


Press and hold button 1, and click button 2 immediately, the current function will be exited.

2.3.Tunnel light

Click button 1 slowly to choose Tunnel light function.

Double click button 1 quickly to enter the function of controlling tunnel light. Then click button 2 to change the color of it, as shown below.



Press and hold button 1, and click button 2 immediately, the current function will be exited.

2.4.Music

Click button 1 slowly to choose Music function.

Double click button 1 quickly to enter the function of controlling music. Then click button 2 to make the buzzer to play a song Happy Birthday.

Press and hold button 1, and click button 2 immediately, the current function will be exited.

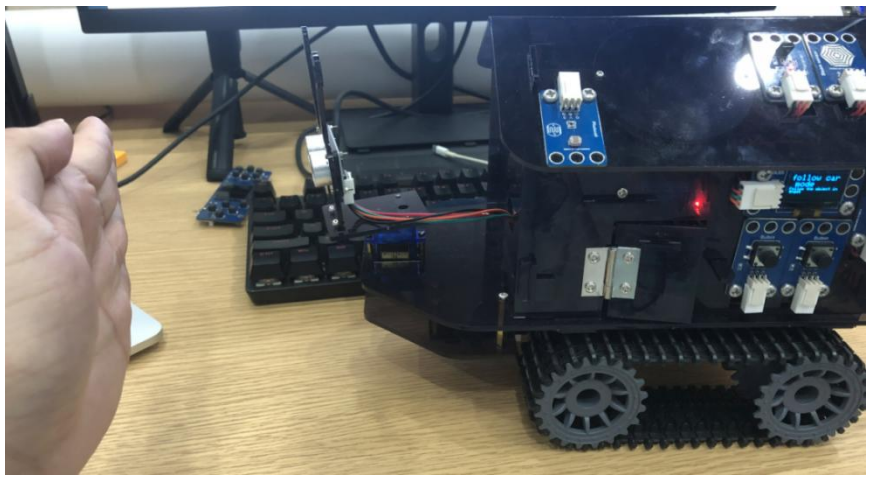
2.5. Follow car

Ultrasonic follow function

Click button 1 slowly to choose Follow car function,as shown below.

Double click button 1 quickly to enter Follow car function.

Then, put your hand or a flat object in front of the ultrasonic sensor on the head of the little snail, and slowly move back and forth, the snail will move back and forth.



Press and hold button 1, and click button 2 immediately, the current function will be exited.

2.6. Avoid Car

Ultrasonic avoidance function

Click button 1 slowly to choose Avoid car function.

Double click button 1 quickly to enter Avoid car function. Once the function is entered, the little snail will move immediately. So, be careful not to let fall off the table.

When the ultrasonic module on the head of Stray Snail judges that the distance between the obstacles in front is close, it will stop. Then the steering gear rotates and the module judges the distance between the obstacles on the left and right sides. The little snail chooses the side with a relatively long distance to turn and move forward.

Press and hold button 1, and click button 2 immediately, the current function will be exited.

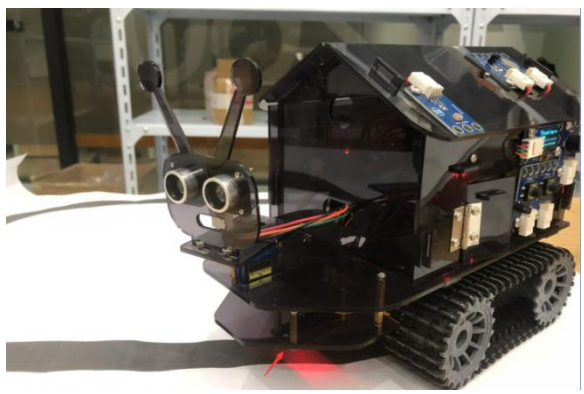
2.7.Tracking Car

Tracking function

Click button 1 slowly to choose Tracking car function.

Double click button 1 quickly to enter Tracking car function.

Take out the tracking drawing provided by us and put Stray Snail on the tracking drawing. When the tracking sensor is aligned with the black track, the little snail will follow the black line. See the figure below.



Press and hold button 1, and click button 2 immediately, the current function will be exited.

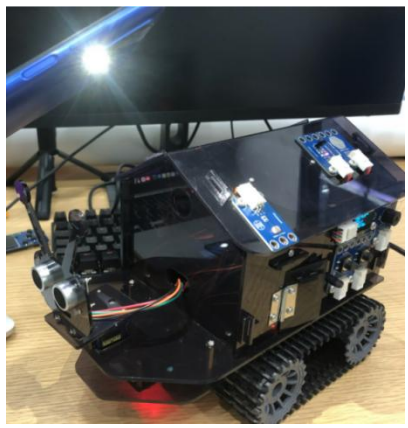
2.8. Find car light

Light control little snail

Note: It needs to be in a dark environment, otherwise the ambient light intensity is greater than the set value, and the little snail will move directly without using a flashlight.

Click button 1 slowly to choose Find car light function, and then double click button 1 quickly to enter it.

Normally, the little snail will not move, and then turn on the flashlight of the mobile phone to irradiate the photosensitive sensors on both sides of the roof. The little snail moves forward. Irradiate the photosensitive sensor on the left and the little snail turns left. Irradiate the photosensitive sensor on the right, and the little snail turns right.



Press and hold button 1, and click button 2 immediately, the current function will be exited.

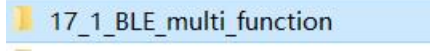
17.Bluetooth APP controls multi-function snail

Write a multi-functional code to control switching functions through APP.

17.1 APP control multi-function

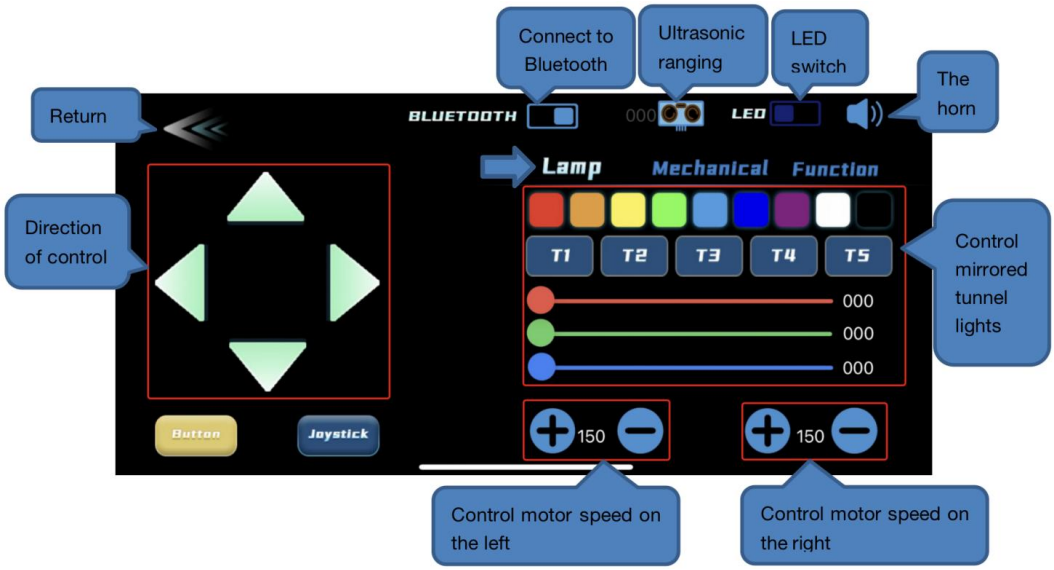
(1)Example code

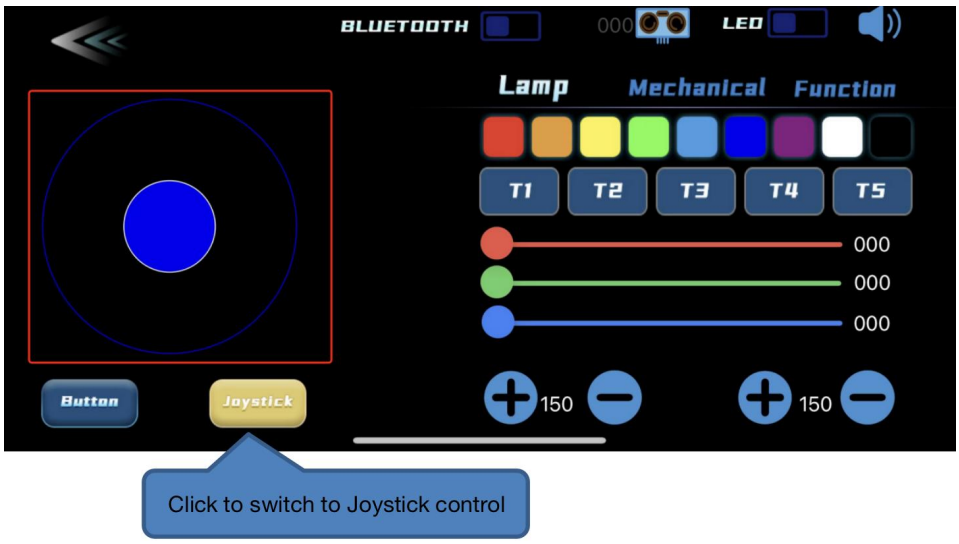
The code is long. Open the example code “17_1_BLE_multi_function” provided by us by using the Arduino IDE.

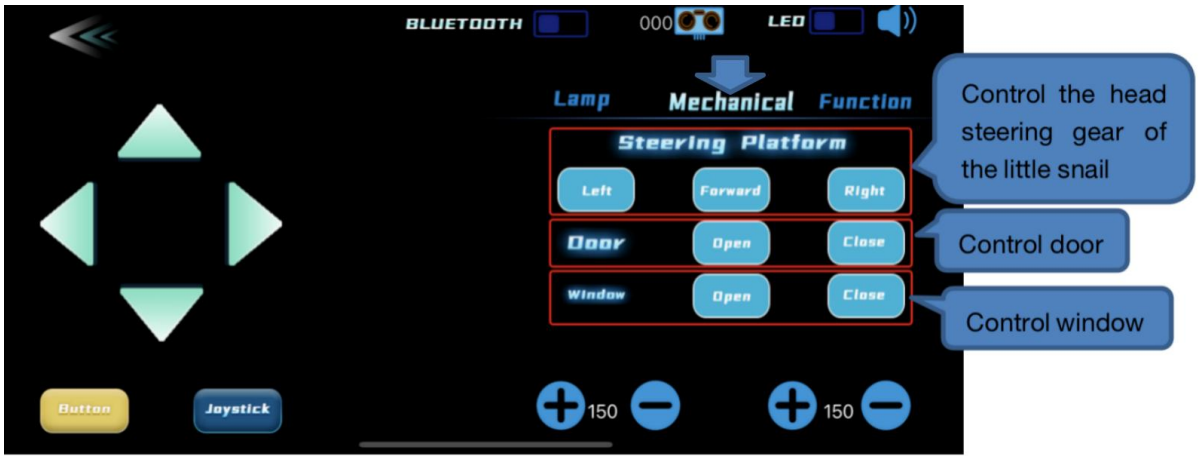


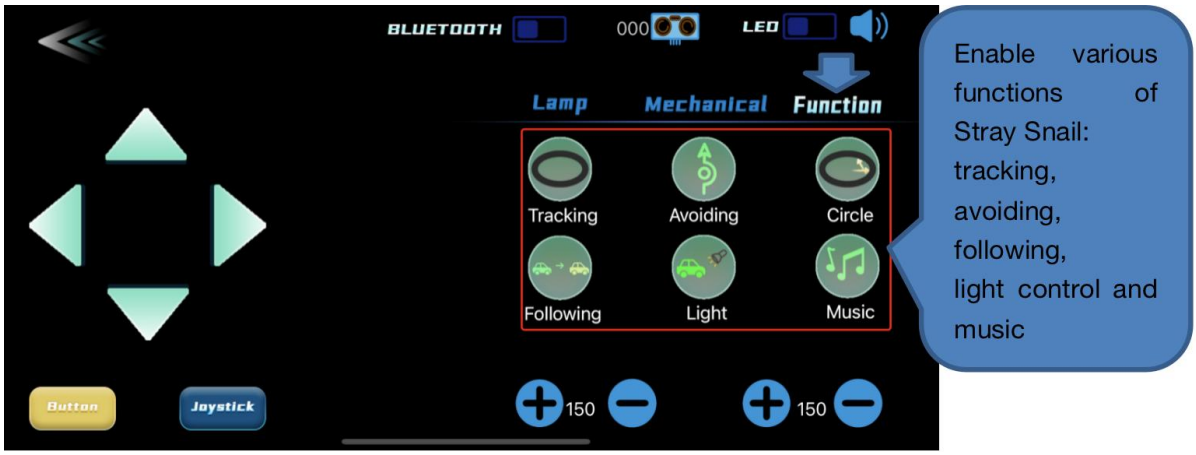
(2)Experiment operation

Operate according to the following figures:


Connect to Bluetooth
Ultrasonic ranging
LED switch
The horn
Return
BLUETOOTH
000
LED
Lamp
Mechanical
Function
Direction of control
T1
T2
T3
T4
T5
Control mirrored tunnel lights
000
000
000
Button
Joystick
150
150
Control motor speed on the left
Control motor speed on the right


BLUETOOTH
000
LED
Lamp
Mechanical
Function
T1
T2
T3
T4
T5
000
000
000
Button
Joystick
150
150
Click to switch to Joystick control


BLUETOOTH
000
LED
Lamp
Mechanical
Function
Steering Platform
Left
Forward
Right
Door
Open
Close
Window
Open
Close
Control the head steering gear of the little snail
Control door
Control window
Button
Joystick
150
150


BLUETOOTH
000
LED
Lamp
Mechanical
Function
Tracking
Avoiding
Circle
Following
Light
Music
Button
Joystick
150
150
Enable various functions of Stray Snail: tracking, avoiding, following, light control and music